Memory for Task-Action Mappings: Mnemonics, Regularity, and Consistency

Adrienne Y. Lee, Peter W. Foltz
and Peter G. Polson

Institute of Cognitive Science
University of Colorado
Boulder, CO 80309

Technical Report 92-11

Abstract

Much of the knowledge required to use modern computing systems takes the form of mappings or associations. These occur between user goals and the functions that accomplish corresponding goals, between functions and the user actions that activate a desired function, and between a menu item or a button label and the function associated with that item or label. The question we explore in this paper is when is it worthwhile, if ever, for a user to pay the price of learning a set of new, regular task-action mappings that is inconsistent with the original set of task-action mappings of a previous version of a system. We consider three factors that determine the ease of learning and retention of task-action mappings: mnemonics, regularity within a set of mappings, and consistency of mapping across different system contexts. In two experiments, we found that Irregular-Non-Mnemonic mappings take much longer to master than Regular-Mnemonic mappings and that Irregular-Non-Mnemonic mappings are more rapidly forgotten and subject to interference effects due to inconsistency. Regular-Non-Mnemonic mappings fall in-between the two groups: 1) they are easier to learn than Irregular-Non-Mnemonic but harder than Regular-Mnemonic mappings and 2) more items are retained for them than the Irregular-Non-Mnemonic mappings but fewer than for the Regular-Mnemonic mappings. We conclude that learning a new set Regular-Mnemonic task-action mappings is worthwhile even at the price of having to unlearn a well-learned set of old task-action mappings.

Memory For Command Sets:
Mnemonics, Regularity, And Consistency

1. Introduction

Much of the knowledge required to use modern computing systems takes the form of mappings or associations. These associations include mappings between a user's goal and the function(s) that accomplish that goal, between a function and the user action(s) that activate that function, and between a menu item or a button label and the function associated with that item or label. Modern application programs are complex (Fischer, 1987, Fischer, Lemke, Mastaglio, & Morch, 1991), having many such mappings. Learning and retention of these mappings is a serious problem for users of modern high performance systems (Engelbeck & Polson, 1988; Norman, 1981, 1988). Many of these mappings are infrequently used and thus are quickly forgotten. In addition, many of the mappings are neither meaningful nor mnemonic and thus are hard to memorize and also quickly forgotten. For example, in UNIX the command "GREP" is used to find a pattern in a file or in the standard input; "GREP" is a mnemonic for "global regular expression print". However, those words have very little meaning for most users, and the labels "find" or "search" might even be more appropriate and/or meaningful.

1.1 THE DEMANDS PUT ON MEMORY BY MODERN APPLICATIONS

Modern graphically oriented (point and click) systems like the Macintosh or Microsoft Windows 3.1 do not avoid the problems of learning and retention of such mappings. For example, the task of saving a copy of a file and at the same time changing its name is accomplished by selecting the item "Save As" from the "File" menu. Thus, successful completion of tasks still requires the user to understand, recognize, and associate the appropriate label to an action sequence that will accomplish a user's current goal (Barnard & Grudin, 1988).

In addition, many menu-based systems have command accelerators that permit experienced users to rapidly execute functions requiring complex menu sequences by using the keyboard. Thus, a user who wants to increase productivity must memorize associations between the function and keystroke sequences. The critical point to be emphasized is that mappings take their prototypical form in command-line based systems like DOS or UNIX, but users of modern graphical user interfaces have to be able to understand and retain a large number of mappings too.

Modern application programs are evolving rapidly. For example, compare the functionality of the original version of MacWrite delivered with the first Macintosh to modern word-processing systems like Microsoft Word 5.0 or MacWrite Pro. This rapid growth of functionality requires that there be a parallel increase in the number of task-action mappings. These sets of mappings are not well structured often having confusing interrelationships between related commands (e.g., EMACS).

1.2 GOALS OF THIS PAPER

The question we explore is: when is it worthwhile, if ever, for users to pay the price of learning a set of new, regular task-action mappings that is

inconsistent with the original set of task-action mappings of a previous version of a system. This paper reports experimental work showing the circumstances under which it is worth learning a new systematically designed set of commands that more adequately capture the expanded functionality of a highly evolved system.

This paper also describes the factors that determine rates of acquisition and probability of retention for mappings or mappings between tasks and commands, functions and user actions that activate those functions, and the labels on menu items and buttons and functions actually performed by the computer. To simplify the discussion, we will refer to such associations as task-action mappings.

1.3 EASE OF LEARNING AND RETENTION OF TASK-ACTION MAPPINGS

Ease of learning and retention of task-action mappings are critical to successful use of a modern complex application. Many factors determine the ease of learning and retention of task-actions mappings (Barnard & Grudin, 1988). The three investigated in this paper are mnemonics, regularity within a set of mappings, and consistency across different system contexts in mappings.

1.3.1 Mnemonics

The term mnemonics refers to the relationship between a specific task description and its associated action sequence. Barnard and Grudin (1988) present an extensive review of this literature and show that most mnemonic systems are some scheme that permits the user to use a straightforward rule such as choose the first letter or vowel deletion that to derive the correct action sequence given knowledge of the proper task label. Examples of mnemonic mappings are found in command accelerators such as the Macintosh Finder. Some fraction of these associations are mnemonic in that keystroke sequence for the accelerator is just the "Command" key plus the first letter of the menu item. Examples include "print", "save", and "quit" on the "File" menu. Furthermore, these commands are both mnemonic and consistent in that the same associations are found in all Macintosh applications.

However, designers soon run out of mnemonic mappings, and thus many of the task-action mappings for Macintosh applications do not follow the simple rule of "Command" shift key plus the first letter of the menu item. Menu items on the "Edit" menu like "Cut" and "Paste" are activated by Command-x and by Command-v respectively. While these bindings are non-mnemonic, they are consistent across all Macintosh applications, and they are used frequently.

1.3.2 Regularity

Regularity is a property of a set of tasks, reflecting a systematic relationship within those tasks. A small set of rules enable users to derive the associated action sequences from descriptions of the tasks (Payne & Green, 1986). We define regularity in the following example that is similar to a subset of the commands in the vi screen editor that is part of the standard distribution of UNIX. A simple task is an editing command represented as an editing operation like insert, delete, copy, and move following by an object like character, word, phrase, and so on. Every task is described by a possible combination of one of the operations followed by one of the objects.

The action sequence associated with each task is the first letter of the operation followed by the first letter of the object. For example, delete word maps onto "DW". Payne and Green developed an attribute grammar formalism that

provides an especially concise representation of such regular sets of task-action mappings. Their formalism requires a small set of rules to describe such a collection of task-action mappings. They present evidence supporting the conjecture that the ease of learning of the set of task-action mappings would be a function of the size of the rule set in their formalism.

Taking this simple text editing example, we have combined two aspects distinguished in our analysis: regularity and mnemonics. Regularity refers to the fact that the task structure, operation-object, is preserved in the structure of the action sequence required to activate the function. The first component of the command designates the operation followed by the component designating the object (delete character, delete word, copy character, copy word, etc.). The mnemonic nature of this command set refers to a second property. There is a single rule that enables you to generate the operation and object designators; the action sequence is just the first letter of the operation followed by the first letter of the object. For example, delete word maps onto "DW". We will use the term "irregular" to refer to a set of task-action mappings that does not preserve the combinatorial operation-object structure illustrated in our example. Thus, a set of task-action mappings that map each of our text editor operator-object pairs onto an arbitrary sequence of letters would be both irregular and non-mnemonic.

Microsoft Word 5.0 has illustrations of regular sets of task-action mappings where the prefix indicates the class of mapping and then the letter indicates a specific command. For example, if the user would like to change the style of some text, these actions define a set of task descriptions that involve selecting the text and then performing actions that are related. The prefix "Command-shift" plus a letter indicates style and paragraph formatting commands. "Command-shift" plus: "B" is mapped to bold, "U" is mapped to underline, "I" is mapped to italics, "C" is mapped to center text, and "Q" is mapped to symbol font. The same letters can be used repeatedly which enables the definition of more mnemonic mappings that are also regular.

1.3.3 Consistency

Another very powerful determinant of the ease with which task-action mappings can be remembered is consistency of use across different applications. For example, in Microsoft Word 5.0, "Command-E" inserts a footnote, whereas in Microsoft Excel 3.0 "Command-E" extracts information from a database. These kinds of inconsistencies have been studied extensively in the human memory literature and they lead to reductions in accuracy of recall and increases in difficulty of learning (Postman, 1971).

The term interference refers to the consequences for ease of learning and successful retention of action sequences in different contexts, that is, the pairing of some task or subtask with different action sequences in different contexts. Consistency is used to refer to the impact on retention and learning of the consistent pairing of a task-action mappings across all contexts for a given system environment.

The research on human learning and memory has made it clear that the effects of these variables can produce large differences in learning rates and in ease of retention. These effects become even more extreme for large sets of task-action mappings that are required by modern application programs such as Lotus 1-2-3 or Microsoft Word 5.0. The critical thing to realize is that any given small piece of functionality in one of these complex systems' effective use requires

knowledge of several such mappings for each function. For example, users may need to know mappings from a variety of tasks to the action sequence for a specific function or mappings from the menu item or a button label to function actually performed by the application.

What characterizes the trade-offs between these various dimensions? Modern applications like high-function word processors have evolved over the years and, in the process of their evolution, often unavoidably introduce various kinds of inconsistencies in the structure of their command sets due to the evolutionary process. Thus, we would like to know when it is worthwhile from the user's point of view to purposefully introduce a massive set of inconsistencies by redesigning the whole command set. One possible time when this re-design may be worthwhile is when a new command set has the goal of introducing a more globally consistent structure in a set of action labels and associated task-action sequences. The surprising result of these studies is that global consistency is always worthwhile even at the price of having to unlearn a well-learned set of old task-action mappings and acquire the new ones.

1.4 RESEARCH ON REGULARITY AND MNEMONICS

Green and Payne (1984) compared the learning of two experimental text-editing command sets. The original command set they used from a commercially-available text editor had conflicting organizational principles. Subsets of the commands are regular in the sense defined in the preceding paragraphs (see section 1.3.2). Some of the commands were partially mnemonic. Payne and Green (1986) constructed a non-mnemonic regular set of key bindings that has a more compact description in the TAG formalism than to the original command. They found that the Regular-Non-Mnemonic command set was easier to learn.

Green and Payne's experiments were the starting point for Walker and Olson's (1988) design of a new command set for a multi-media, document editor called Diamond. Diamond permits the construction of multi-media documents that incorporate text, object-oriented graphics, bit-mapped graphics, and spreadsheets. The original set of key bindings for Diamond were irregular and partially mnemonic. They were derived from EMACS. Walker and Olson constructed a new set of key bindings that were both regular and mnemonic. Tasks were characterized on three dimensions: context (system, text editor, bit mapped graphics editor, etc.), editing action, and editing object. The action sequence for a single command indicated context, action, and object. On the structured recall, there was no significant difference between groups; but, on the randomly ordered list recall, the regular-mnemonic command set was much better recalled than the partially mnemonic-irregular command set. However, the recall rate for both editors on the random list recall was very low.

Green and Payne (1984), Payne (1985), and Payne and Green (1986) make the point that regular command sets can be extended in an orderly fashion. The designer can define new actions, new objects and the individual user actions that designate the components of a task. The user with knowledge of this structure would be able to learn easily the new commands and even predict what a novel, unknown command would be. Reisner (1981) has shown that if commands have simple generation rules, that learning and transfer are strongly facilitated.

Evidence from verbal learning experiments can be found in Foss (1968) where he showed that subjects are easily able to extend analogs of regular task-action mappings. He trained subjects in an experimental paradigm originally developed by Esper (1925). The stimuli were generated by the twenty-five possible combinations of five different shapes and five different colors. The required responses were two nonsense syllables with five nonsense syllables consistently associated with each of the colors and another set of five nonsense syllables associated with each of the shapes. Foss trained subjects on a subset of the total of twenty-five mappings that would enable them to infer the underlying rules the five nonsense syllables associated with each shape combined with the five nonsense syllables associated with each color. They were then transferred to the remaining novel color shape combinations and showed a very high level of performance, demonstrating that subjects can extract from the training such underlying rules. When Foss introduced a small number of inconsistencies into the complete set of twenty-five mappings, he found that the inconsistencies were disruptive; subjects treated the items as individual associations. Subjects did not learn the rules that correctly described a majority of the mappings.

1.5 TASK-ACTION MAPPINGS, TRANSFER, AND VERBAL LEARNING.

The results of the learning of regular task-action mappings is only part of the information we need to decide whether one should change the mappings for an existing application to a set of new regular task-action mappings. There is the possibility that the old, well-learned task-action mappings may interfere with the acquisition of the new regular mappings. If the old task-action mappings intrude, they would contribute to higher error rates on the new application and might be executed in the context defined by the new version of the application.

Learning of associations that are analogous to task-action mappings has been studied in the human learning and memory literature for over a hundred years (Ebbinghaus, 1885; Postman, 1971). Transfer of training, i.e., whether knowledge of the old mappings will interfere or facilitate with acquisition of the new mappings, is a major issue that has been studied in the verbal learning literature under the name, interference theory (Postman, 1971). In these experiments, the material to be learned were collections of stimulus-response pairs.

In the human-computer action domain, we claim that one can make the analogy by identifying the stimuli with task descriptions and the responses with the user actions required to perform that task. The notation used in the verbal learning literature to describe the situation of learning a new regular command set is a transfer experiment of the form A-B to A-C, where A is the collection of elementary tasks common to both old and new versions of the system, and B and C are the different action sequences associated with the old, irregular command set and the new, regular command set, respectively.

A-B, A-C has been shown to be a negative transfer paradigm under a fairly broad range of experimental conditions (Martin, 1965; Postman, 1971). By negative transfer we mean that when performance is compared to a control condition in which subjects learn unrelated lists of the form A-B, D-E (where D-E indicate that the second list is completely unrelated to the first) one finds that learning and retention of A-B interferes with the learning and retention of A-C when use performance on D-E is used a base-line. Negative transfer describes the

empirical phenomena in which the performance of an experimental group is inferior to an appropriate control. Inhibition and interference are used both as theoretical and empirical constructs to characterize the reduction in retention when compared to an appropriate control. The literature uses negative transfer, interference, and recall as complementary manifestations of some common underlying process (McGeoch, 1942), although such theoretical interpretations are controversial.

In the experiments reported in this paper, we examine two kinds of interference relevant to different situations, proactive interference and retroactive interference. Proactive interference refers to interference from old skills on the acquisition and retention of new skills. In addition, we consider a one-shot transfer, the situation where one is going from an old version of the system to a new version without returning to the original version. Proactive interference an occur in these situations of one-shot transfer where old task-action mappings can interfere with the new task-action mappings.

Many types of interference possibilities exist, especially in a mixed vendor environment. One would want to be sure that a new skill does not interfere with existing skills. Retroactive and proactive interference are both possible. Retroactive interference refers to the interference in retention and performance of old skills by newly-learned skills. Much of the verbal learning literature has tested alternative explanations of various interference effects in both learning and retention (Postman, 1971).

1.6 SUMMARY

In summary, task-action mappings in their prototypical form of task sets and related key bindings can be equated to lists of stimulus-response pairs used in important segments of research on human learning and memory starting with Ebbinghaus in the late 19th Century. We are proposing to use the experimental methodology developed in this literature to study potential sources of negative transfer in proactive and retroactive inhibition when people learn multiple related bits of task-action mappings. We also hope to find that mnemonic task-action mapping will facilitate learning because knowledge of the task enables one to generate the action sequence which dramatically facilitates memorization of the action sequence. Thus, regularity defines two sets of structures. The first set of structures describes the interrelationship between a collection of tasks in terms of values on a collection of two or more dimensions like action and object. The given task is defined by a combination of these dimensions. The second set of structures is the systematic mapping from these task features onto the action sequence. This mapping may be either mnemonic or non-mnemonic.

1.7 DESCRIPTION OF CURRENT RESEARCH

This paper reports two experiments in the learning and retention of task-action mappings under conditions that permit us to evaluate both negative transfer and proactive and retroactive interference. The first study investigated learning and retention of actual text editor commands in situations where subjects were trained to a relatively high level of performance before being transferred to a second editor or tested for retention. This study investigated the interactions of two different command sets. The first was a Regular-Mnemonic command set. The second was an Irregular-Non-Mnemonic command set. The second study, following Walker and Olson (1988) treated the learning of task-

action mappings as a paired-associate learning task. Subjects had to learn, in an anticipation paradigm, associations between task descriptions like delete character with the correct action either DC or control-Z. The second experiment also evaluated the contribution of mnemonic and non-mnemonic mappings in regular command sets.

2. Experiment 1

In Experiment 1, subjects in the experimental groups learned two editors that had identical sets of editing functions which were activated by different action sequences. One editor had Irregular-Non-Mnemonic task-action mappings (IRNM). The second had Regular-Mnemonic task-action mappings (RM). The goal of this experiment was to simultaneously investigate the impact of the mnemonic in regularity with potential interference effects caused by the inconsistencies in the two command sets. Note that in this experiment we are looking at the impact of learning different kinds of task-action mappings in the context of an actual editing task. A formal analysis of the knowledge necessary to perform the editing task like those done by Bovair, Kieras, and Polson (1990) which show that the task-action mappings are a small part of the knowledge necessary to edit text.

2.1 METHOD

2.1.1 Subjects

Thirty four subjects with minimal to no computer background participated in this experiment. The data of four subjects was discarded because they did not complete the training. Twelve subjects were recruited from introductory psychology courses and twelve subjects were hired through a newspaper ad.

2.1.2 Design

The design of the experiment is shown in Table 1. The experimental groups (1, 2, 4, and 5) learned the two different editors in each possible training order and then for each initial learning training order were tested for their retention of the commands in each of the two different possible test orders, tasks 5 and 6. The two control groups, 3 and 6, each learned a single editor. Subjects were randomly assigned to one of the six experimental conditions on order of their appearance in the laboratory.

Table 1
Summary of conditions for experiment 1

	Day 1			Day 2	
Group	Task 1	Task 2	Task 3	Task 4	Task 5
1	Learn RM	Learn IRNM	Recall Both	Test RM	Test IRNM
2	Learn RM	Learn IRNM	Recall Both	Test IRNM	Test RM
3	Learn RM		Recall RM	Test RM	
4	Learn IRNM	Learn RM	Recall Both	Test RM	Test IRNM
5	Learn IRNM	Learn RM	Recall Both	Test IRNM	Test RM
6	Learn IRNM		Recall INM	Test IRNM	

2.1.3 Apparatus

The experiment was executed under the control of a DEC Micro VAX II running two interacting programs, a computer assisted instruction (CAI) system

and a text editor. There were two DEC VT 220 character oriented displays on a desk in front of the subject. Instructions and feedback was presented on one display under control of the CAI system; the editing task were presented and performed on the other terminal. The CAI system controlled the overall procedure, presented instructions and tutorial material to subjects, and evaluated the subjects' responses. The other program was especially designed editor, construction tool that enabled us to implement the experimental editors and was controlled by subject input or by commands from the CAI system.

Table 2
Summary of commands for experiment 1

Commands	Mnemonic	Non-mnemonic
Delete Character	DC	ctrl-K
Delete Word	DW	ctrl-Z
Delete Phrase	DP	ctrl-Q
Delete Sentence	DS	ctrl-O
Delete Line	DL	ctrl-P
Copy Character	CC	ctrl-T
Copy Word	CW	ctrl-D
Copy Phrase	CP	ctrl-V
Copy Sentence	CS	ctrl-N
Copy Line	CL	ctrl-G
Transpose Character	TC	ctrl-J
Transpose Word	TW	ctrl-X
Transpose Phrase	TP	ctrl-E
Transpose Sentence	TS	ctrl-R
Transpose Line	TL	ctrl-L

The commands for the two experimenters are presented in Table 2. Both had an identical overall structure with 15 editing operations defined by the combination of three editing functions (delete, copy, and transpose) and 5 text objects (character, word, line, phrase, and sentence). All cursor positioning was done with arrow keys. There were no "find" or other location commands. The only difference between the two editors were the action sequences used to activate the individual editing functions. The RM editor's mappings were generated from the first letter of the editing function and the first letter of the text object (e.g. copy line maps onto CL). The IRNM editor action sequences combined the control shift key with a letter that had no connection with either the function or the object.

The texts to be edited were presented in booklets. The editing function was indicated in the right hand margin; the text object to be edited was highlighted in the text. The texts were taken from a draft of an introductory history text book and had been reverse edited so that when the correct editing operation was applied, the resulting text would be correct.

The overall structure of individual editing operations was identical for both editors. Each editing function began with a subject using the cursor keys to position the cursor on the first letter of the object to be edited. The subject then issued the appropriate editing command defined by the combination of editing

operation and text object. For the delete command, the specified object was removed from the text. The copy and transpose commands required subjects to use the arrow keys to position the cursor at the point where material was to be inserted or on the first character of the second object that was to be transposed. Pressing the ENTER key caused the copy or transpose function to take place. The subject then had to confirm the editing operation completed by either pressing the ENTER key or pressing an UNDO key. The UNDO key undid the effects of the last command and repositioned the cursor at the end of the last successful edit. The ENTER key caused the text to be sent to the CAI package for evaluation.

2.2 PROCEDURE

The experiment took place over two days. On the first day, subjects learned one or two editors. On the second day, they were tested for retention of one or two command sets. The same training procedure was used for all six groups and on the two editors learned by the experimental groups. There was a ten minute break between training sessions for the experimental group.

2.2.1 Day 1

The training process for a given editor involved three phases. In the first phase, subjects received identical introductory material that covered the process of text editing, use of the cursor keys, and a description of the notation used to specify edits. The material was presented, one screen at a time, by the CAI system; subjects had complete control over the presentation rate and could review previously presented screens.

During the second phase, subjects went through a series of three training episodes one for each editing function. They first received instruction on how to do a given editing operation on a word and then had to successfully complete five edits using the command that they had just studied. The editing operations were tutored in the order delete, copy, and then transpose. Subjects were given detailed descriptions of how to do the command including cursor positioning and the use of the ENTER and UNDO keys. Subjects were required to do each edit in the order that it appeared in the text and to avoid skipping any edits since the skipped edit was treated as an error. On a given edit after the first error, the CAI system provided a minimal amount of feedback. If the subject immediately made a second error, the system provided a detailed description of how to perform the edit correctly. After either minimal or detailed feedback on an error, the text editor was reset and the subjects had to redo the erroneous edit. The subjects had to complete five edits without any errors before they were allowed to go on to the material on the next command or the next phase of training.

During the third training phase, subjects first studied all 15 task-action mappings shown in Table 2, and then edited manuscripts containing edits that require that they use all 15 task-action mappings. During the study component of this phase, they were shown a sequence of three screens, each screen containing the five commands associated with one editing operation like delete. Subjects could study each screen for as long as they wish and could review previously presented screens.

Subjects were then required to perform an editing task in which they had read a text which contained in a random order a block of 15 edits defined by all possible combinations of editing operations and objects. The order of the 15 possible edits was randomized. Subjects were given the correct command after

making an error and had to redo the incorrect edit. The criterion for learning was one error-free execution of all 15 edits. The amount of time it took a subject to reach criterion on the editing task and the number of times through the manuscript were our primary measures of the difficulty of learning the task-action mappings for the two different editors.

2.2.2 Day 2

On Day 2, subjects were given two retention tests. On the first test, subjects were shown the editing commands one at a time, such as delete word, on flash cards and were asked to respond verbally with the keystroke sequence and editor. Subjects were encouraged to attempt to recall the action sequences for both editors. Commands were presented in the different random order for each subject. No feedback was provided during testing. However, subjects were in the same experimental room as where they learned one or two text editors and could use the keyboard to assist in their recall of the current keystroke sequences.

In the verbal learning literature, this test paradigm is called modified, modified, free recall (Barnes & Underwood, 1958). This paradigm provides a very sensitive test for estimating their overall retention and for measuring potential retroactive or proactive interference effects *without* artificially restricting subjects to: 1) recalling the command sequences in any particular order and 2) only forcing them to recall one or the other command of the task-action mappings.

The second recall test required subjects to edit a one page manuscript with 15 edits using one or both editor(s) that they had learned previously. Subjects were required to repeat the task until they had successfully editing executed all 15 edits without error. If they made a mistake they were provided with feedback on the correct task action mapping. The order of testing on two editors was counterbalanced for the experimental subjects and there was no rest between the two editing sessions.

2.3 RESULTS

First, we report the learning data for the first editor aggregated into two groups, (1-3 and 4-6) comparing results on the acquisition of the Regular-Mnemonic (RM) and the Irregular-Non-Mnemonic (IRNM) task-action mappings. Next we will briefly discuss the second editor learning data. Finally, we will present an analysis of the Day 2 retention data.

2.3.1 Acquisition of the first editor

Our analyses focused on the third phase training where subjects studied 15 task-action mappings and then did an editing task to a criterion of one errorless repetition of all edits. During the first two phases, subjects were learning the overall structure of the text editing task, how to do cursor positioning, how to interpret a marked up manuscript, and acquiring other skills necessary to use a text editor. These sets of skills were identical for the two editors. By the time subjects started the third phase of the training procedure, mastering the task-action mappings was the one missing component of the complete set of skills needed to edit a manuscript. Thus, subjects' performance in this phase is primarily determined by the difficulty of mastering the task-action mappings.

The mean times and standard deviations (in parenthesis) to complete the third phase for either one or two editors are presented in Table 3. Combining the data for the experimental and control groups on the first editor gives the mean time to complete the edits for the Regular-Mnemonic editor as 14.52 (8.73) min.

and the mean time to complete the edits for the Irregular-Non-Mnemonic editor as 58.47 (22.62) min.; the three to one difference in training time is highly significant, $F(1,28) = 77.33$, $p<.01$.

Table 3

Means and standard deviations (in parentheses) of the total time in min. to complete the third training phase for one or two editors

Group (n)	**1st editor**	Time (min.)	**2nd editor**	Time (min.)
Experimental (10)	IRNM	68.7 (23.1)	RM	14.6 (8.24)
Control (5)	IRNM	38.0 (21.7)		
Experimental (10)	RM	11.6 (8.03)	IRNM	35.2 (10.7)
Control (5)	RM	20.3 (10.0)		

We also did an analysis of the cycles to reach criterion where a cycle is one pass through the 15 edits defined by the combinations of three editing operations and five objects. The means and standard deviations (in parentheses) are show in Table 4. Combining the data for the experimental and control groups on the first editor gives the mean number of cycles to complete the edits for the Regular-Mnemonic editor as 1.7 (0.58) cycles and the mean number of cycles to complete the editors for the Irregular-Non-Mnemonic editor as 3.3 (1.07) cycles; the difference in cycles is highly significant, $F(1,18) = 7.63$, $p<.01$.

Table 4

Means and standard deviations (in parentheses) of cycles required to reach criterion on the third training phase for one or two editors

Group (n)	**1st editor**	Cycles	**2nd editor**	Cycles
Experimental (10)	IRNM	3.3 (1.0)	RM	1.9 (0.9)
Control (5)	IRNM	3.2 (1.3)		
Experimental (10)	RM	1.6 (0.5)	IRNM	2.9 (0.7)
Control (5)	RM	2.0 (0.7)		

Table 5 shows the percent recalled on the second day. For those subjects who only learned a single editor, the controls, a greater percentage was remembered by the Regular-Mnemonic subjects, $F(1,8)=6.46$, $p<.05$. For those subjects who learned two editors, the Regular-Mnemonic editor commands were better remembered whether the command set was learned first or second, $F(1,18)=17.04$, $p<.01$ and $F(1,18)=14.17$, $p<.01$.

Table 5

Means and standard deviations (in parentheses) of percent correct on Day 2 recall task

Group (n)	1st editor	% recall	2nd editor	% recall
Experimental (10)	IRNM	61.3 (27.4)	RM	94.0 (11.5)
Control (5)	IRNM	48.0 (39.3)		
Experimental (10)	RM	97.3 (3.44)	IRNM	48.0 (36.9)
Control (5)	RM	94.7 (11.9)		

Table 6 shows the amount of time that subjects took to complete the editing task. For control subjects, no difference was measured, although the Regular-Mnemonic condition was slightly faster $F(1,8)=3.05$, $p<.1$. For subjects learning two editors, editing with the Regular-Mnemonic editor was uniformly faster whether the editor was used first or second, $F(1,18)=33.91$, $p<.01$ and $F(1,18)=6.98$, $p<.01$.

Table 6

Means and standard deviations (in parentheses) of time to complete Day 2 editing task.

Group (n)	First editor	Time (min.)	Second editor	Time (min.)
Experimental (10)	Non-Mnemonic	13.28 (4.66)	Mnemonic	5.63 (3.61)
Control (5)	Non-Mnemonic	12.4 (15.42)		
Experimental (10)	Mnemonic	3.26 (2.81)	Non-Mnemonic	11.69 (6.30)
Control (5)	Mnemonic	2.15 (0.84)		

2.3 DISCUSSION

We focus on the three major results that were found in Experiment 1. First, Regular-Mnemonic task-action mappings were dramatically easier to learn than the Irregular-Non-Mnemonic. There was an almost four to one difference in timed criterion (see Table 3) for the third training phase where subjects had to memorize 15 task-action mappings and then use the editor to edit a one page manuscript that contained edits requiring each possible combination of editing commands and text objects.

The recall and skill retention data collected on Day 2 also show dramatic differences between the two editors. Recall of subjects for the commands of the Regular-Mnemonic editor were nearly perfect and there was no suggestion of proactive or retroactive interference in the recall data for this command set. The recall data show that retroactive inhibition occurred when subjects first learned the Irregular-Non-Mnemonic editor and then were required to learn the Regular-Mnemonic editor.

The pattern of recall results and the fact that a significant percentage of the subjects using the Regular-Mnemonic editor made no errors during the third phase in training or in recall suggests that they were using the regular structure to generate the correct commands for novel editing operations leading to perfect performance during training and would enable them to generate the correct answer when they could not directly retrieve the proper combination from long

term memory. These results clearly show large differences in favor of a Regular-Mnemonic command set.

2.3.1 Training day results

The training day results indicated that the Irregular-Non-Mnemonic editor was more difficult to learn than the Regular-Mnemonic editor. However, even though subjects took longer to learn the Irregular-Non-mnemonic editor, the number of times through the 15 command-response editor task was the same as for both editors. For those who learned two editors, the means in Table 3 indicated that the amount of time to learn the second editor after learning the first editor (either mnemonic or non-mnemonic) was shorter than the original learning time for the subjects who had to learn that second editor first. These results indicate that some amount of general text editor transfer exists (such as how to position the cursor) (see Polson, Bovair & Kieras, 1987; Singley & Anderson, 1985, 1989).

2.3.2 Interference effects

Retroactive inhibition was expected for the Irregular-Non-Mnemonic editor learned first but not for the Regular-Mnemonic editor. The data indicated retroactive inhibition existed for the Irregular-Non-Mnemonic editor but was not found for the Regular-Mnemonic editor. These results indicate that subjects were using a rule structure when learning the Regular-Mnemonic editor first. Proactive inhibition was not indicated for either condition (mnemonic or non-mnemonic). Several possible reasons could account for these findings. First, the proactive inhibition may have been just too small to measure, as when the lists are overlearned. Second, response meaningfulness prevented the mnemonic commands from being forgotten or from experiencing interference from the second set.

2.3.3 Summary of Experiment 1

The results obtained in this experiment are complementary to the findings of Walker and Olson (1988). In particular, this experiment demonstrated that Walker and Olson results can carry over to real text editors. Recall rates were higher than those found by Walker and Olson. This result may be based on the fact that subjects were required to use the actual text editor and therefore the commands had a higher meaningfulness to the subjects and there were fewer commands to learn. In addition, subjects who learned the Regular-Mnemonic command set learned more quickly and remembered more than those subjects who learned the Irregular-Non-Mnemonic command set. For both the Walker and Olson and this experiment, the Irregular-Non-Mnemonic command set was compared with a Regular-Mnemonic command set. Although we found differences between these two sets, it is not clear whether it was the mnemonics or the regularity that made the difference in the learning, recall and transfer.

3. Experiment 2

The primary goal of Experiment 2 was to separate the contributions of regularity and mnemonics on speed of learning and probability of retaining task-action mappings. In addition, the study also evaluate the impact of inconsistencies between two task-action mappings acquired one after the other. Cramer (1990) showed that mnemonics, with a very simple and small set of rules, was effective in promoting later recall. Payne and Green (1986) focused on a theoretical characterization of regularity using an attribute grammar. Regular but non-mnemonics mappings between task description reaction sequence

require the introduction of secondary rules to describe the arbitrary relationship between a task feature and the corresponding action sequence. Payne and Green argued that these secondary rules would have less impact on the learning and that regularity was the major determinant of the rate of learning.

In this experiment, we reduced the problem of learning task-action mappings to a simple associative learning task. Unlike Experiment 1 where subjects had to perform the actual editing functions, subjects in this experiment learned paired-associates where the stimuli were 15 editing task labels made up of a command-object combination and the responses were the action sequences associated with each editing task. The paired associate learning task is the essence of the third phase of the training process for both editors used in Experiment 1. Recall was the subject's primary task. The procedure we used in this study was very similar to that used by Walker and Olson (1988) in their experiment comparing an Irregular-Non-Mnemonic command set with a Regular-Mnemonic command set.

3.1 METHOD

3.1.1 Subjects

Eighty eight subjects with minimal or no computing background participated in this experiment. They were recruited from Introductory Psychology classes and served in the experiment in partial fulfillment of a course requirement. Ten subjects served as pilot subjects and 12 were discarded because they did not complete both days of the experiment leaving 66 subjects who completed both days.

3.1.2 Materials

The materials were four lists of paired-associates. Three of the lists represented task-action mappings. The stimuli were the 15 combinations of editing operations (delete, copy, transpose) combined with five objects (character, word, phrase, line, and sentence). The three sets of responses for the three paired-associate list are shown in Table 7. The first was Regular-Mnemonic (RM) which was identical to the task-action mappings for the Regular-Mnemonic editor used in Experiment 1. The second set was Regular-Non-Mnemonic (RNM); objects and actions were consistently paired to unrelated letters with three editing operations and five text objects. As shown in Table 7, copy character was MP, while transpose character was YP. So the letter P represents the object "character". The third set was the Irregular-Non-Mnemonic (IRNM) task-action mappings where the editing tasks were paired with single unrelated letters combined with the control shift key (as in Experiment 1).

The fourth list of non-specific, paired associates was used as an additional control condition (NS). The stimuli were common high frequency nouns (non-computer terms, paired with two digit numbers). This condition was included because it permitted comparison of a completely meaningless list of paired associates to the acquisition of different sets of meaningful-task-action mappings.

3.1.3 Design

Subjects were randomly assigned in order of appearance in the laboratory to one of ten conditions shown in Table 8. Subjects in the four control conditions learned just one of the four lists described in the preceding material sections. Subjects in the remaining six experimental groups learned two out of three of the paired associate lists that were defined by one of the task-action mappings.

Table 7
Task-action mappings for Experiment 2

Editing Tasks	Action Sequence		
	RM	RNM	IRNM
Delete Character	DC	SP	ctrl-K
Delete Word	DW	SC	ctrl-Z
Delete Phrase	DP	SF	ctrl-Q
Delete Sentence	DS	SI	ctrl-O
Delete Line	DL	SE	ctrl-P
Copy Character	CC	MP	ctrl-T
Copy Word	CW	MC	ctrl-D
Copy Phrase	CP	MF	ctrl-V
Copy Sentence	CS	MI	ctrl-N
Copy Line	CL	ME	ctrl-G
Transpose Character	TC	YP	ctrl-J
Transpose Word	TW	YC	ctrl-X
Transpose Phrase	TP	YF	ctrl-E
Transpose Sentence	TS	YI	ctrl-R
Transpose Line	TL	YE	ctrl-L

Table 8
Conditions for experiment 2

	Day 1		Day 2
Group	Task 1	Task 2	Task
1	Learn RM	Learn RNM	Verbal Recall
2	Learn RM	Learn IRNM	Verbal Recall
3	Learn RM		Verbal Recall
4	Learn RNM	Learn RM	Verbal Recall
5	Learn RNM	Learn IRNM	Verbal Recall
6	Learn RNM		Verbal Recall
7	Learn IRNM	Learn RM	Verbal Recall
8	Learn IRNM	Learn RNM	Verbal Recall
9	Learn IRNM		Verbal Recall
10	Learn non-specific		Verbal Recall

3.1.4 Procedure

The experimental procedure took two successive days. On the first day, subjects learned one or two paired associate lists depending upon their experimental condition. On the second day, they were tested for retention of the associations (task-action mappings) learned on Day 1. Table 7 shows the list learned by each group. Subjects in the fourth control group learned one set of paired associates (task-action mapping). The experimental group learned two sets of task-action mappings.

A common training procedure was used for all lists. Subjects learned each list to a criterion of one perfect recitation using the anticipation procedure. Subjects were seated at a computer terminal and told that they would have to

learn a set of either associates or command-response pairs. The commands would appear on the screen and the subjects would have to type in the appropriate action sequence. When an error was made, the subject was given feedback on the correct action sequence or response. The list of pairs was randomized before each presentation. There was a brief pause between the acquisition of successive lists for subjects in the experimental group.

On Day 2, subjects were tested in the same experimental room facing the experimenter while seated near the computer and its keyboard. The experimenter read either the stimulus words or an editing command-object pair and subjects were asked to recall one or both responses that had been associated with the word or task description. Experimental subjects were allowed to recall the responses in either order after list presentation of the editing command-object pair. Subjects were also permitted to recall all responses out of order at a later time if they wished.

3.2 RESULTS

We present the results for Experiment 2 in three sub-sections. First, we will describe the effects of the four different kinds of materials on first list learning. Second, we test if possible interference effects exist on the second list learning processes due to relationships between the first and second lists in the six experimental groups. Finally, we will present the Day 2 recall data. Here our focus is on possible interference effects on retention of both first and second list mappings.

Performance by the NS group was not significantly different from the IRNM group in both learning and recall. This indicates that performance on the IRNM list is similar to a random list of paired associates. Thus, their means will be included for comparison but no further discussion of these results will be made here.

Table 9

Means and standard deviations (in parentheses) of the total time in min. to reach criterion on each paired association list for all groups and lists.

Group (n)	First List	Time (min.)	Second List	Time (min.)
1 (10)	RM	4.3 (1.80)	RNM	13.3 (5.8))
2 (6)	RM	8.9 (15.6)	IRNM	31.9 (15.9)
3 (5)	RM	4.9 (2.20)	None	
4 (10)	RNM	19.0 (6.60)	RM	3.6 (0.3)
5 (5)	RNM	14.5 (3.90)	IRNM	38.4 (7.3)
6 (8)	RNM	19.4 (7.50)	None	
7 (6)	IRNM	37.9 (10.2)	RM	2.9 (1.0)
8 (4)	IRNM	43.2 (5.70)	RNM	7.3 (3.6)
9 (6)	IRNM	56.0 (10.2)	None	
10 (6)	NS	29.3 (15.4)		

Note: Regular-Mnemonic (RM), Regular-Non-Mnemonic (RNM), Irregular-Non-Mnemonic (IRNM)

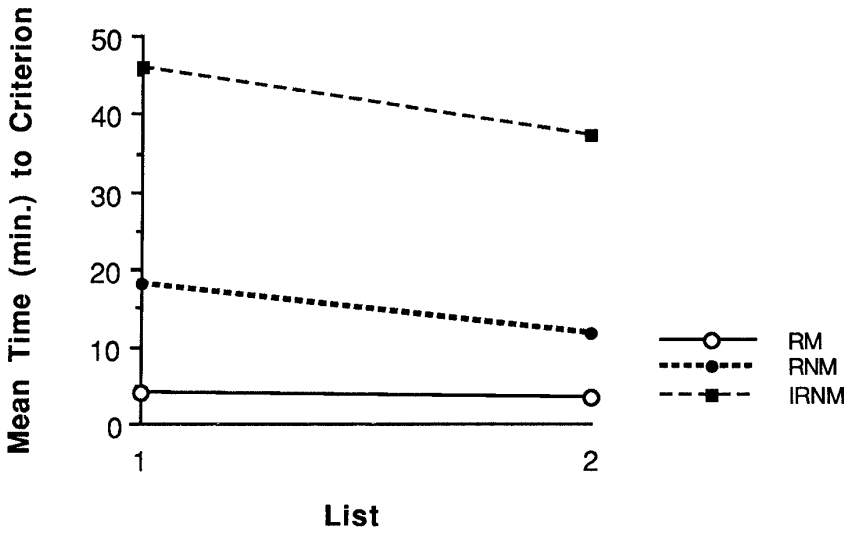

Figure 1. Mean time to criterion in minutes plotted as a function of list for Regular-Mnemonic (RM), Regular-Non-Mnemonic (RNM), Irregular-Non-Mnemonic (IRNM) mappings.

3.2.1 First List Learning

Table 9 contains the mean times in minutes to reach criterion and the standard deviations (in parentheses) for all conditions and both lists for the experimental groups. Figure 1 shows the data aggregated across the various groups by task-action mapping learned on the first or second list. Thus, the data point in the lower left-hand corner of the figure is the average time to reach criterion for all the groups who learned the Regular-Mnemonic task-action mappings for list one, representing the 21 subjects in Groups 1 through 3. The data point in the lower right hand corner is the average time for subjects in Groups 4 and 7, 14 subjects who learned the Regular-Mnemonic editor on group 2. No single analysis of variance would be appropriate to evaluate simultaneously the data contained in Table 9 and Figure 1. Thus, we did several different independent analyses.

The differences in List 1 performance shown in Figure 1 were evaluated by doing a one-way ANOVA of the List 1 times to criterion for the first 9 groups shown in Table 9. The differences between the means were highly significant (RM recalled more than RNM, $F(1,42)=2.14$, $p<.05$ and RM recalled more than IRNM, $F(1,36)=9.7$, $p<.05$); post-hoc tests evaluating differences between all possible pairs of means using the Tukey-A procedures showed that these differences were highly significant, too. Identical analyses were done on the List 2 data plotted in Figure 1 from groups 1, 2, 4, 5, 7, and 8 shown in Table 9.

Practice effects were evaluated by performing a two-way analysis of variance treating the data from first and second lists of the six experimental as if

they were from independent groups. The main effect due to differences in task-action mappings was highly significant ($F(2,98)=242.8$, $p<.01$) as was the main effect due to differences between Lists 1 and 2 ($F(1,99)=14.6$, $p<.01$). There was no significant interaction supporting the impression given by Figure 1.

3.2.2 Possible Interference Effect On List 2 Learning

Testing for possible proactive interference effects on List 2 learning was done using independent groups t-test to compare appropriate second List times to criterion with relevant control groups. Observe that these comparisons have much less power because we are only using subsets of the total collection of data shown in Table 9. We performed three sets of two t-tests to evaluate whether learning a given task action mapping was slowed by first learning one of the other two mappings. In general, we would expect that second list learning for a given type of task-action mapping would be faster because of the large generalized learning-to-learn effects found in paired associate learning.

The second List Regular-Mnemonic (RM) mappings were learned significantly faster than the List 1 RM control. ($t(13)=2.0$, $p<.05$ and $t(9)=3.0$, $p<.05$) Similar results where found for the learning of Irregular-Non-Mnemonic (IRNM) mappings after either RM or RNM lists ($t(10)=4.0$, $p<.01$ and $t(9)=3.9$, $p<.01$). When comparing Regular-Non-Mnemonic (RNM) mappings learned second after RM, it takes less time to learn RNM ($t(16)=3.0$, $p<.01$). However, RNM was not facilitated by learning IRNM first. Thus, in most cases, learning was facilitated by learning another command set before.

3.2.3 One Trial Learning

One feature of the RM mappings is that only one rule is needed in order to master the complete set of key bindings. A subject only needs to learn that the first letter of a text editing command corresponds to the first letter of the action and the second letter of the action correspond to the first letter of the object. Of the subjects learning the RM editor, 16% made no errors on their first learning cycle and thus guessed the correct rule. An additional 39% made a single error on their first trial but then no errors on the other trials on that cycle. With one piece of feedback, subjects were able to infer the correct rule and made no additional errors. Thus, a Regular-Mnemonic editor affords one trial learning for over 50% of the subjects.

Table 10

Means and standard deviations (in parentheses) of percent recalled on Day 2 for each paired associate list for all groups and lists.

Group (n)	First List	% Recalled	Second List	% Recalled
1 (10)	RM	100.0 (0.0)	RNM	94.0 (9.30)
2 (6)	RM	98.0 (3.0)	IRNM	52.0 (12.7)
3 (5)	RM	100.0 (0.0)	None	
4 (10)	RNM	86.0 (20.0)	RM	99.3 (2.00)
5 (5)	RNM	59.3 (29.3)	IRNM	78.7 (12.7)
6 (8)	RNM	94.0 (11.3)	None	
7 (6)	IRNM	45.3 (32.7)	RM	97.8 (3.50)
8 (4)	IRNM	76.7 (8.70)	RNM	80.0 (14.7)
9 (6)	IRNM	76.7 (15.3)	None	
10 (6)	NS	76.7 (14.7)		

Note: Regular-Mnemonic (RM), Regular-Non-Mnemonic (RNM), Irregular-Non-Mnemonic (IRNM)

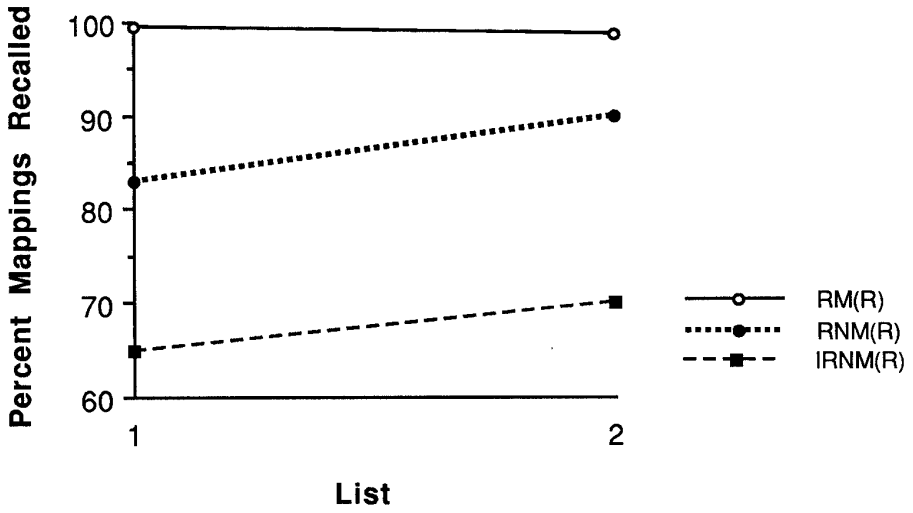

Figure 2. Mean percentage recall plotted as a function of list for Regular-Mnemonic (RM), Regular-Non-Mnemonic (RNM), Irregular-Non-Mnemonic (IRNM) mappings.

3.2.4 Day 2 Recall

Recall data are shown in Table 10 and Figure 2. We use an identical sequences of statistical test of analyze the recall data. Table 10 contains the mean percentages of mappings recalled and the standard deviations (in parentheses) for all conditions and both lists for the experimental groups. Figure 2 shows the

recall data aggregated across the various groups by task-action mapping learned on the first or second list.

List 1 and 2 recall performance plotted in Figure 2 was evaluated using two one-way ANOVAs on the recall data for List 1 from the first 9 groups shown in Table 10 and on the List 2 data from groups 1,2 4, 5, 7, and 8 shown in Table 10. Separate sets of post-hoc tests for each List evaluating differences between all possible pairs of means using the Tukey-A procedures showed that these differences were highly significant, too. For controls, more items were recalled for the RN and RNM sets than for the IRNM or NS sets, $F(1,23)=17.4$, $p<.01$ Comparing the recall for the RM, RNM, and IRNM sets learned first, more commands were remembered for RM than RNM or IRNM ($F(1,42)=2.1$, $p<.05$ and $F(1,36)=9.7$, $p<.05$).

Practice or possible interference effects were evaluated by performing a two-way analysis of variance treating the data from first and second lists of the six experimental as if they were from independent groups. The main effect due to differences in task-action mappings was highly significant ($F(2,76)=20.2$, $p<01$). There was no difference due to List and no significant interaction supporting the impression given by Figure 2. The Regular-Mnemonic task-action mappings are easiest to remember, followed by the Regular-Non-Mnemonic and then finally the Irregular-Non-Mnemonic mappings.

3.2.5 Retroactive interference

Testing for possible retroactive interference effects on List 2 recall was done using independent groups t-test to compare the first task action mapping learned with its appropriate control. When the IRNM mapping is learned first and the RM mapping is learned second, fewer IRNM actions are remembered than the IRNM control, $t(10)=3.33$, $p<.05$. This indicates that subjects had difficulty remembering the IRNM command set after learning a far easier command set, the RM command set. When the RNM is learned first and the IRNM is learned second, fewer RNM commands are remembered than the RNM control, $t(11)=3.51$, $p<.05$. This indicates that the RNM command set was difficult to remember when the IRNM command set was learned after it.

3.2.6 Proactive interference

For proactive interference, retention of the second task-action mapping learned is compared with retention performance of its control. When the RM mappings are learned first and the IRNM mappings are learned 2nd, fewer IRNM actions are remembered than the IRNM control, $t(10)=2.6$, $p<.05$. This indicates that the RM mappings interfered with the recall of the IRNM mappings. When IRNM mappings are learned first and RNM mappings are learned 2nd, fewer RNM commands are remembered than the control, $t(8)= 3.22$, $p<.05$. This indicates that the IRNM mappings interfered with the learning of the RNM mappings. These results are interesting considering that when the RM mappings were learned after the IRNM set, no proactive interference was found. Since the RNM set requires more rules to remember, some memory limitation could possibly be reached.

3.3 DISCUSSION

In Experiment 1, the learning of Regular-Mnemonic and Irregular-Non-Mnemonic task-action mappings was evaluated in the context of actual use of a text editor. Experiment 2 distilled the question of learning task-action mappings

down to its essence by having subjects learn paired-associate lists defined by the mappings. The results of Experiment 2 showed a vast superiority for Regular-Mnemonic mappings. They were learned very rapidly, in a single presentation by a majority of subjects, and were very resistant to both forgetting and interference.

The performance of the Regular-Non-Mnemonic group was intermediate to that of the two extremes - the Regular-Mnemonic and the Irregular-Non-Mnemonic groups. In other words, having to learn the arbitrary mappings between an editing action and a keystroke and a text object and another keystroke extracted a price. Furthermore, these arbitrary associations were subject to interference.

Comparing the performance of the Irregular-Non-Mnemonic groups with the other two clearly shows that regularity affects performance above and beyond that of mnemonics. The data are very nicely accounted for by the insights provided by Payne and Green. In their analysis, the Regular-Mnemonic group essentially has to learn a single rule that decodes the editing command-editing object representation and maps it using the first-letter mnemonic onto the correct task-action sequence. In this experiment, the Regular-Non-Mnemonic group has to know nine rules; one for each type of edit (delete, copy, transpose), one for each type of object (character, word, phrase, sentence, line) and one to combine the type of edit and object. On the other hand, for the Irregular-Non-Mnemonic group, each command is an arbitrarily chosen letter combined with the single action to use the CTRL key. Thus, they must learn the 15 separate commands independent of any generalizable rule. This idea is confirmed by the comparable performance of the non-specific group which learned words paired with numbers (paired-associate learning task).

4. Conclusion

Both experiments, one embedding the learning of task-action mappings in the acquisition of an actual text editor, and the other, distilling the learning of task-action mappings to its essence by using a paired-associate learning methodology, gave consistent results. Irregular-Non-Mnemonic mappings are much harder to learn, and more difficult to remember than Regular-Mnemonic task-action mappings. Furthermore, non-mnemonic mappings are subjected to both proactive and retroactive interference effects found in the paired associate learning literature.

These conclusions summarize a significant portion of a hundred years of research on paired-associate learning. Arbitrary associations are difficult to learn and even more difficult to remember. Furthermore, they are subject to both retroactive and proactive interference effects. Regularity and mnemonic mappings impose regularities on the structure of the material to be memorized, making it more meaningful. Again, the paired-associate learning literature and the memory literature show that the human memory system is very sensitive to such manipulations and structured, meaningful material is vastly easier to memorize than arbitrary relationships.

People who learn arbitrary associations can later retrieve them reliably. The price they have to pay is extensive practice. Designers have to be aware that frequency of use for various commands, the opportunity for practice, follows a relationship like Zipf's law. That is, a small subset of the commands are used very frequently and the large majority of commands are used infrequently.

Infrequently used, arbitrary task-action mappings are a source of real difficulties because they are easily forgotten and subjected to the kinds of interference effects demonstrated in this study.

The original question we asked in our introduction was: is it ever worth it to have a designer completely revamp an interface, restructuring the task-action mappings and changing all the commands? This can be answered with an unequivocal "yes". However, there are important qualifications. Consider the task used in this experiment, text editing. Part of the skills of using an editor involve decomposing the task of modifying a manuscript into a sequence of simple editing actions that operating on single text objects. This simple task can be performed by a single command activated by a sequence of user actions. The results presented in this paper presuppose that a user decomposes the original editing task into a sequence of simple tasks each represented as an editing function on a text object. The regular mapping takes as input the correct representation of the function to be performed and generates as output the action sequence that performs that function. In order to gain the benefits of regular task-action mappings shown in Experiment 2, one has to describe the functionality of the system in terms that have the appropriate structure.

In conclusion, Regular-Mnemonic task-action mappings convey dramatic advantages over other types of task-action mappings. They can be learned in a single exposure to a small subset of the commands. In addition, our results strongly suggest that the learning of a new set of Regular-Mnemonic task-action mappings would not be interfered with by knowledge of the mappings required by the previous version of the interface. Furthermore, the representations of these mappings would not be interfered with by inconsistent mappings used in other applications. Thus, although a dramatic re-design to the interface, such as replacing all Irregular-Non-Mnemonic task-action mappings with Regular-Mnemonic ones, may entail a great deal of effort, the benefit to the user may outweigh the costs to the designer.

Acknowledgements

The authors acknowledge Tektronix which supported the first author through a corporate gift in the work on the first experiment. Further, the ARI grant MDA 903-89-D-0025 supported the authors in the final statistical analysis and writing of this paper. The opinions expressed in this paper are those of the authors and not necessarily those of any of the supporting organizations.

Special thanks to Tom Mix for helping run a good portion of the subjects. Also, thanks to Susan Davies, Nancy Pennington, Paula Messamer, and Robert Nicolich for helpful discussions on this research.

References

Barnard, P., & Grudin, J. (1988) Command languages. In M. Helander (Ed.), *Handbook of Human-Computer Interaction.* (pp. 237-256). Amsterdam: North-Holland Press.

Barnes, J.M., & Underwood, B.J. (1959). "Fate" of first list associations in transfer theory. *Journal of Experimental Psychology, 62*, 152-170.

Bovair, S., Kieras, D.E., & Polson, P.G. (1990). The acquisition and performance of text-editing skill: A cognitive complexity analysis. *Human-Computer Interaction, 5*(1), 1-48.

Cramer, M. (1990). Structure and mnemonics in computer and command languages. *International Journal of Man-Machine Studies, 32*, 707-722.

Ebbinghaus, H. (1885). *Memory: A contribution to experimental psychology* (translated by H.A. Ruger & C.E. Bussenves, 1913). NY: Teacher's College, Columbia University.

Engelbeck, G., & Polson, P.G. (1987). An interference theory explanation of retention of errors. *SIGCHI Bulletin, 19*(3), 61-63

Esper, E.A. (1925). A technique for the experimental investigation of associative interference in artificial linguistic material. *Language Monograph, 1.*

Fischer, G. (1987) Making Computers more Useful and more Usable. *Proceedings of the 2nd International Conference on Human-Computer Interaction* (Honolulu, Hawaii). (pp. 97-104). New York: Elsevier Science Publishers.

Fischer, G., Lemke, A. C., Mastaglio, T., & Morch,A. (1991) The Role of Critiquing in Cooperative Problem Solving, *ACM Transactions on Information Systems, 9*(2), 123-151.

Foss, D.J. (1968). An analysis of learning in a miniature linguistic system. *Journal of Experimental Psychology, 76*(3), 450-459.

Green, T.R.G., & Payne, S.J. (1984). Organization and learnability of computer languages. *International Journal of Man-Machine Studies, 21*, 7-18.

McGeoch, J.A. (1942). *The psychology of human learning.* New York: David McKay Company.

Martin, E. (1965). Transfer of verbal paired associates. *Psychological Review, 72*, 327-343.

Norman, D.A. (1981). The trouble with UNIX. *Datamation, 27*(12), 139-150.

Norman, D.A. (1988). *The psychology of everyday things.* NY: Basic Books, Inc.

Payne, S.J. (1985). Task-action grammars: The mental representation of task languages in human-computer interaction. Unpublished doctoral dissertation, University of Sheffield.

Payne, S.J., & Green, T.R.G. (1986). Task-action grammars: A model of the mental representation of task languages. *Human-Computer Interaction, 2*, 93-133.

Polson, P.G., Bovair, S., & Kieras, D.E. (1987). Transfer between text editors. In J.M. Carroll & P. Tanner (Eds.), *Proceedings of CHI '87 Human Factors in Computing Systems and Graphics Interface Conference*. (pp. 27-32). New York: Association for Computing Machinery.

Postman, L. (1971). Transfer, interference, and forgetting. In J.W. Kling & L. A. Riggs (Eds.), *Experimental Psychology*. (pp. 1019-1032). New York, NY: Holt, Rinehart, and Winston, Inc.

Postman, L., Stark, K., & Fraser, J. (1968). Temporal changes in interference. *Journal of Verbal Learning and Verbal Behavior, 7*, 672-694.

Postman, L., & Underwood, B.J. (1973). Critical issues in interface theory. *Memory and Cognition, 1*, 19-40.

Reisner, P. (1981). Formal grammar and human factors design of an interactive graphics system. *IEEE Transactions of Software Engineering, SE-7*, 229, 240.

Sheffield, F. D. (1946). The role of meaningfulness of stimulus and response in verbal learning. Ph.D. dissertation, Yale University.

Singley, M.K., & Anderson, J.R. (1985). The transfer of text-editing skill. *Journal of Man-Machine Studies, 22*, 403-423.

Singley, M.K., & Anderson, J.R. (1989). *The transfer of cognitive skill.* Cambridge, MA: Harvard University Press.

Walker, N., & Olson, J.R. (1988). Designing keybindings to be easy to learn and resistant to forgetting even when the set of commands is large. In E. Soloway, D. Frye & S.B. Sheppard (Eds.), *Human factors in computing systems, CHI '88*. (pp. 201-206). New York: Association for Computing Machinery, Inc.