CIS-TR 85-11

Orthogonality in the Design of Command Languages

Cathryn A. Stanford, Edward M. Bowden**, David Locke*, and Sarah A. Douglas**

*Department of Computer and Information Science
**Department of Psychology
University of Oregon

Orthogonality in the Design of Command Languages

Cathryn A. Stanford[*], Edward M. Bowden[**], David Locke[*], and Sarah A. Douglas[*]

[*]Department of Computer and Information Sciences
[**]Department of Psychology
University of Oregon
Eugene, Oregon 97403

ABSTRACT

Research has shown that organization plays an important role in memory. The present study applies these findings to the design of a command language. The concept of orthogonality is used to maximize the internal organization of a text editing command language. This orthogonal language is compared to an organized, but non-orthogonal, and an anti-organized language on measures of predictability, recall, and performance. Subjects in the orthogonal language condition performed better than subjects in the other conditions on all measures. General steps necessary to design an orthogonal language are discussed.

INTRODUCTION

As computers become commonplace, the needs and desires of novice and casual users must be given greater consideration. One way to make computer use simpler is to develop command languages that are powerful, yet easy to learn and to use. Included in the growing body of work concerned with the design of command languages are studies involving the selection of mnemonic symbols (Perlman, 1984), suggestiveness of command names (Black and Moran, 1982; Rosenberg, 1982), the use of icons (Hemenway, 1982), and menu design (Card, 1982; Perlman, 1981).

Such work presents some promising suggestions, but has had disappointingly little impact on the design of actual command languages. Studies to date have concentrated on guidelines for the selection of "good" command names, while neglecting the importance of the language's overall organization.

Psychological research has consistently shown that organization enhances memory performance. Decades ago psychologists demonstrated that organized stimuli are more easily learned and more accurately recalled than stimuli lacking organization. Subjects were found to have a strong tendency to recall lists of related, but randomly presented, stimuli in an organized fashion; grouping related stimuli such as animal names together at recall (Bousfield, 1953; Cohen and Bousfield, 1956; Jenkins and Russell, 1952). Further studies found a direct relation between the degree of objective categorical organization and the amount subjects later recalled (Bousfield, Cohen and Whitmarsh, 1958; Jenkins, Mink and Russell, 1958; Sakado, 1956).

Tulving (1962) extended the work on organizational processes to stimuli that had no apparent a priori organization and found that subjects tended to impose a sequential structure on their recall when organization seemed otherwise lacking. He also found that memory performance improved as the degree of subjective organization increased. This suggests that people have a tendency to impose structure on their environment and use that structure to enhance memory.

There is further evidence, from the areas of concept formation and rule induction, that stimuli designed to reflect rules can be learned very rapidly (Jones and Zamostny, 1975; Kotovsky and Simon, 1973; Restle, 1970; Restle, 1976; Restle and Brown, 1970) suggesting that people are very sensitive to regularities in the presentation of stimuli. Further, intentional learning of objectively organized material is often no better than incidental learning (Mandler, 1967; Orstein, Trabasso and Johnson-Laird, 1974) suggesting that organized information can be acquired with minimum effort on the learner's part.

The question of why organization leads to improved memory performance has important implications for the design of command languages. Some theorists (Bower, 1970; Tulving, 1967) suggest that when subjects recognize preexisting organization in the stimuli they extract rules which specify the relations between the various stimuli. At the time of recall these rules serve to restrict the range of response alternatives that must be considered, or help single out the correct response from the set of alternatives. Subjects extract rules that describe the relations between stimuli, then use these rules to calculate or generate the appropriate response. By extracting a general rule the subject no longer need remember each separate stimulus.

Acknowledgements: We would like to thank Joachim Krueger for running subjects in a pilot study. We would also like to thank John Conery, Douglas Hintzman, Steven Keele and Michael Posner for helpful comments on an earlier draft of this paper.

One need only remember the rule to both generate the original set of stimuli, and predict stimuli that were not presented.

Several researchers have recognized the importance of these findings for the design of command languages (Green, 1983; Green and Payne, 1984; Perlman, 1984). Perhaps the most significant of these studies is that of Green and Payne (1984). They suggest that the overall structure of a language is more important than any individual features, and that consistency between the rules of the language should be used as a "guiding principle" in the design of command languages. Languages designed with consistency between the language rules should be easier to learn and remember than languages lacking consistency.

Green and Payne (1984) used four different experimental command languages, each of which was a 26-command subset of a text-editing language. Each of the four languages represented different combinations of organization and use of mnemonics. Language 1 used both a consistent organizing principle and what Green and Payne called "sensible" mnemonics. Language 2 used the same organizing principle, but was non-mnemonic. Language 3 had no overall organizing principle, but used the best mnemonics possible given the constraint of being able to vary only a single symbol in a command name. Language 4 had sensible mnemonics, but contained two conflicting organizing principles. Subjects were given 12 minutes to learn one of these command languages, then were tested with either free or prompted recall.

Green and Payne (1984) found that subjects learning Language 1 recalled significantly more command names than subjects learning any of the other languages, when tested with free recall. They also recalled significantly more command names when tested with prompted recall than did subjects learning Language 3 or Language 4. These results convincingly support their hypothesis that languages exhibiting consistent organization will be easier to learn than languages exhibiting no organization. Green and Payne further suggest that a consistent scheme of organization within the language is more important than the particular names or symbols used to represent any individual command.

The above study has clear implications for the design of command languages. Languages that have a readily visible structure in the form of a consistent syntax should be easier to learn than languages that lack such structure. A command language with conflicting organizational rules will be more difficult to learn than one having no organizational rules, even if the former employes mnemonic command names (Green and Payne, 1984; Perlman, 1984). We find the principle of consistency between language rules to be compelling and we believe that it should be further extended.

The difficulty with which a command language is learned will depend in part upon the number of relevant elements that must be learned, the saliency of these elements, and the complexity of the rules used to combine the elements. If the elements are combined according to a simple and consistent rule, the command set should be relatively easy to learn. The "ideal" command language appears to be one that is designed to contain a restricted set of elements and to reflect consistent rules of syntax. The use of an orthogonal command language appears to be a way to achieve both of these goals.

While the concept of orthogonality is a frequently cited design criterion for computer languages (c.f. Horowitz, 1984; Tanenbaum, 1984; MacLennan, 1983; Meyers, 1978) and is occasionally mentioned in relation to command languages (Green, 1983; Green and Payne, 1984; Perlman, 1984), to the best of our knowledge its oft-cited cognitive advantage to the user has never been empirically tested. Language writers have stressed its usefulness in providing functionality while reducing the level of complexity in computer instruction sets, e.g. the PDP-11. Others emphasize the increase in generality(power) in higher level programming languages, e. g. ALGOL 68, by eliminating restrictions or special cases.

Orthogonal literally means, "composed of right angles" but is more commonly understood to mean that the various components of the whole are independent of each other. Command languages can be broken down into their basic factors--that is, the irreducible information that must be expressed to produce any action. This should be distinguished from the symbols used within individual commands to represent the information contained in each factor domain. In the context of a command language, orthogonality means that each basic factor necessary for the execution of the command should be represented independently of all other basic factors. A change in a single factor should not cause a change in any of the remaining factors. For example, in a command language for text-editing the command for moving the pointer to the left one word should differ from the command to delete to the left one word by only the symbol that distinguishes the DELETE mode from the MOVE mode. Symbols representing other factors should remain unchanged by a change in mode. Thus, an orthogonal command language would require relatively few symbols to produce a vast number of discriminably different commands. This reduction in the overall set of symbols required allows the designer to make best advantage of mnemonics in the selection of command names.

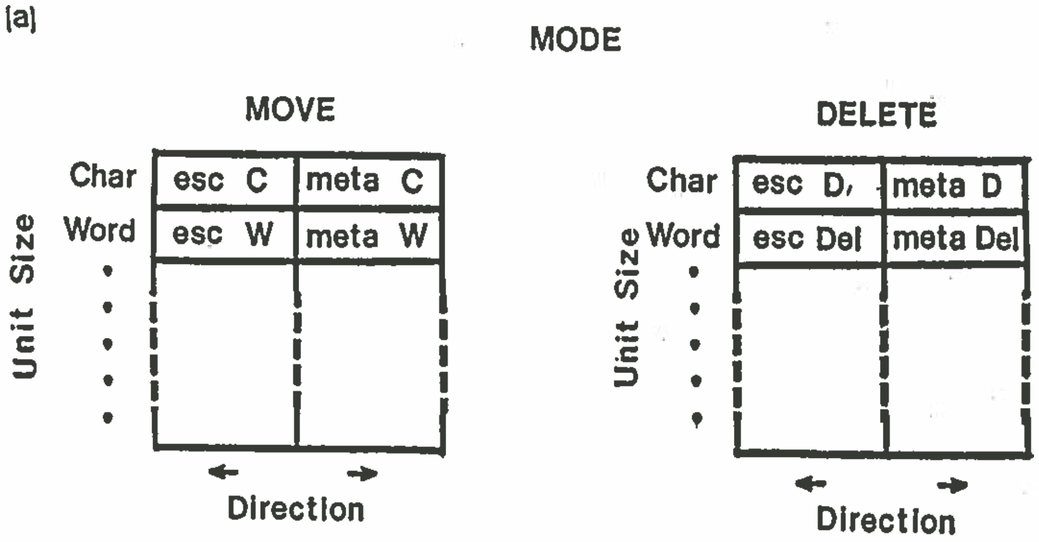
Although a number of researchers (Green, 1983; Green and Payne, 1984; Perlman, 1984) have mentioned the importance of orthogonality in the design of command languages, they appear to have overlooked one vital fact. By definition a command language cannot be orthogonal unless each basic factor is represented independently of all other factors. In the domain of text-editing, most command languages use two symbol commands (i.e. commands requiring two keys to be pressed), yet many actions that one might wish to perform while text-editing consist of three basic factors; those of Mode, Direction, and Unit Size. When any language uses only two symbols in each command to express three factors, one of the symbols will, by necessity, contain information about at least two of the three factors. Therefore, it is impossible to change one factor independently of all other factors. This situation inevitably leads to poor use of mnemonics in the selection of the symbols making up the commands, or, even worse, conflicting organizational rules.

Perhaps the best way to clarify this difference between a command language that is orthogonal and one that is organized, but not orthogonal, is by use of diagrams. A command language that uses two symbols to represent three factors is illustrated in Figure 1a. An orthogonal command language is illustrated in Figure 1b. One indication that the orthogonal command language is more economical is that all three factors can be represented in a single diagram, whereas the non-orthogonal language requires a separate diagram for each change in the factor not explicitly represented on the diagram. Each additional diagram represents an additional rule that the subject must learn.

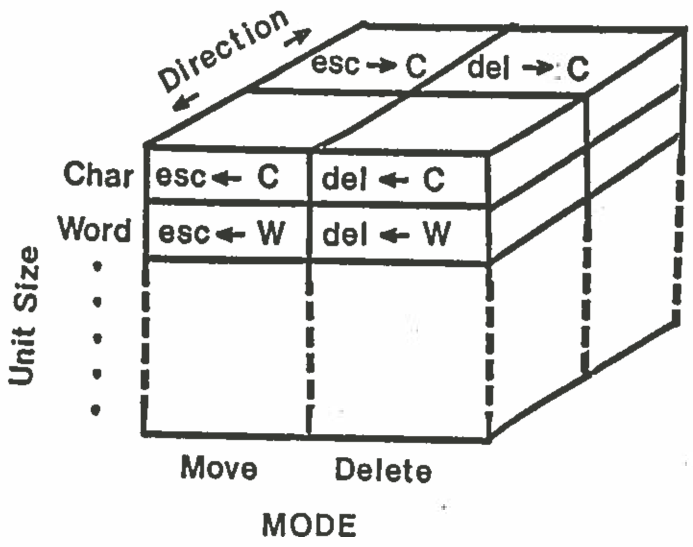
FIGURE 1

Diagrams representing an Organized (a) and an Orthogonal (b) command language.

(a)



(b)



These diagrams also suggest that commands should be more easily generated within an orthogonal command language. A given command does not have to be recognized as a unit, but can be derived from the point of intersection of its various factors.

The economy of an orthogonal language can also be represented with set grammars, developed by Payne and Green (1983). Both the structure and the legal strings of a command language can be seen in a set grammar. Maximum objective organization can be achieved by minimizing the number of rules necessary to define the commands. The fewer rules, the more consistent, and therefore more predictable, the language should be.

Table 1 illustrates that an orthogonal language can be specified with fewer rules than the organized language. In the grammar for the organized language the mode does not have its own symbol, but rather is coupled with the unit size. This requires an additional rule to distinguish the two modes. The user must then know which unit size symbols are used for movement and which are used for deletion. The set grammar also illustrates how a non-orthogonal language is limited in its ability to use mnemonics. As shown in the example, the movement commands for the organized language exhaust all obvious mnemonics for unit size, so that the designer has been left with poorly chosen symbols for deletion commands.

EXPERIMENT 1

Reisner (1981) equates consistency with predictability, therefore a language with consistent rules of organization should be more predictable and more readily learned and remembered than a language with inconsistent rules of organization. We believe that when the factors of a language are orthogonal predictability is maximized. Users will be readily able to extract the syntax of the language as well as the mnemonics used in deriving individual symbols, and to use this knowledge to generate the entire language. Thus, individual commands within an orthogonally organized language should be predictable from knowledge of the underlying rules of organization. Reisner (1981) further suggests that the predictability of commands within a language could be one way to measure the adequacy of the language's design.

In Experiment 1 the predictability of three languages with different levels of organization was examined. Each of the experimental languages was a subset of a text-editing language. The first language was a subset of Emacs, and had little, and often conflicting , organization. This language will be referred to as anti-organized. The second language was taken from Language 1 in Green and Payne (1984). They state that it had a consistent organization and sensible mnemonics. This language will be referred to as organized. The third language was created for this experiment using both an orthogonal organization scheme and sensible mnemonics, and will be referred to as orthogonal. These languages were a subset of those presented in Table 2.

Table 1 : Set grammar description of organized and orthogonal languages.

Organized Languages

Sets

Direction:: (Meta, Esc)
Movement-Unit Size:: (C, W, L, S, [, F)
Deletion-Unit Size:: (D, Del, K, Z)

Rules

Sentence ::= Direction + Movement-Unit Size
Sentence ::= Direction + Deletion-Unit Size

Selection Rule

Free Choice

Orthogonal Language

Sets

Mode:: (Move, Delete)
Direction:: (<--, -->)
Unit Size:: (C, W, L, S, P, F)

Rules

Sentence ::= Mode + Direction + Unit Size

Selection Rule

Free choice

Table 2 : The experimental languages

	Orthogonal	Organized	Anti-organized
move forward a character	esc → C	meta C*	ctrl F
move backward a character	esc ← C	esc C	ctrl B
move forward a word	esc → W	meta W	esc F
move backward a word	esc ← W	esc W	esc B
move forward a line	esc → L	meta L	ctrl E
move backward a line	esc ← L	esc L	ctrl A
move forward a sentence	esc → S	meta S	ctrl (
move backward a sentence	esc ← S	esc S	ctrl)
move forward a paragraph	esc → P	meta [	ctrl [
move backward a paragraph	esc ← P	esc [	ctrl]
move forward a document (file)	esc → D	meta F	ctrl <
move backward a document (file)	esc ← D	esc F	ctrl >
delete forward a character	del → C	meta D	ctrl D
delete backward a character	del ← C	esc D	ctrl H
delete forward a word	del → W	meta del	esc D
delete backward a word	del ← W	esc del	esc H
delete forward a line	del → L	meta K	ctrl K
delete backward a line	del ← L	esc K	ctrl L
delete forward a sentence	del → S	meta Z	ctrl .
delete backward a sentence	del ← S	esc Z	ctrl ,

*Meta, esc, and del keys each required individual keystrokes. Ctrl was pressed simultaneously with its associated key.

Method

Subjects.

Subjects were 30 undergraduates enrolled in various introductory level psychology courses at the University of Oregon. No subjects had previous experience with text-editing languages. All subjects participated for course credit.

Procedure

Subjects were randomly assigned to one of the three experimental languages. These languages consisted of anti-organized, organized and orthogonal languages. Ten subjects were tested with each language. Subjects were tested in groups of 2 to 5. The experimenter gave subjects a brief oral explanation of text-editing in general, and editing commands. Subjects were then given one of three single pages containing a list of 16 text-editing actions (see Appendix A). Two of the actions were paired with their corresponding command names. One command was chosen from each of the modes. The same command-action pairs were given to all subjects. They were instructed to generate appropriate command names for the specified actions using these two command name-action pairs as models. They were allowed as much time as necessary to complete the task. The number of correctly generated command names was recorded for each subject.

Results and Conclusions

Mean number of correctly generated command names, out of 16 possible, was 0.0 (SD = 0.0), 1.3 (SD = 1.57), and 13.4 (SD = 0.84) for the anti-organized, organized, and orthogonal languages, respectively. Because no subject in the anti-organized condition was able to correctly generate a command the assumption of equal variance was violated and statistical analysis seemed inappropriate. The differences in predictability between the three languages are large enough, however, that their significance is apparent.

These results show clearly that an orthogonal command language can be generated from a small knowledge base. Subjects can easily extract the organizational rules in an orthogonal language and then use this knowledge to generate unknown commands. The organized, but non-orthogonal language was far less predictable, whereas predictability of the anti-organized language appeared to be nonexistent.

One concern with this experiment might be that mnemonics were not held constant across languages. Thus, it could reasonably be claimed that the orthogonal language was more predictable because of better use of mnemonics, rather than the greater degree of organization. It is important to realize, however, that the differences in the organization of the languages make holding mnemonics constant impossible. For example, in the organized language given that meta W moves the pointer forward a word, one might expect that W would again be used to designate word when deleting. However, if W is used to designate word in both modes, meta must be changed or the modes will be indistinguishable. This would disrupt the language's organization. Therefore, one cannot attribute the differences in predictability solely to mnemonics. The organizational scheme of an orthogonal language allows it to make better use of mnemonics.

In addition, if differences in mnenomics were the sole factor responsible for differences in subjects' ability to generate commands, one would expect subjects in the organized condition to be able to generate about 50% as many commands as subjects in the orthogonal condition because about 50% of the mnemonics were identical for the two languages. In fact, subjects in the organized condition were able to generate less than 10% as many commands as subjects in the orthogonal condition. Even if problems of conflicting mnemonics or greater memory load, for subjects in the organized condition, are taken into account one would not expect the difference in performance to be so large.

EXPERIMENT 2

The results of Experiment 1 suggest that the more organization there is within the structure of a command language, the more predictable the language will be. It is possible, however, that the differences in predictability obtained were due to inadvertent selection of command name-action pairs that revealed the structure of the orthogonal language, while obscuring the structure of the organized language. If this is true, then providing subjects with the entire set of commands should make any existing organization evident. Knowing the organization of the command language should make the individual commands easier to learn and recall because forgotten or unlearned commands can be generated from knowledge of the language's structure. Therefore, if differences in predictability found in Experiment 1 were due solely to poorly selected examples, they should vanish when the entire language is provided for learning and recall is tested. If an orthogonal language is truly more predictable we would expect to find superior recall for individual commands. The initial part of Experiment 2 was designed to test this prediction.

Researchers in the design of command languages commonly use pen-and-paper tests to measure the effects of manipulations. However, measures of performance are seldom used. The second purpose of Experiment 2 was to test subject's performance, while learning, with the languages in actual text-editing tasks.

Finally, a foreseeable criticism of the orthogonal design is that it requires an additional keystroke to perform many actions, thus requiring additional time to complete editing tasks. The text-editing portion of Experiment 2 was also designed to investigate whether performance, while learning, was slowed by the additional keystroke.

Method

Subjects.

Subjects were 30 undergraduate psychology students at the University of Oregon who participated for course credit. None had previous experience with the use of text-editors.

Materials.

Materials for the recall test consisted of a single page containing a brief description of text-editing commands and a list of 20 command names paired with their actions. The three languages are presented in Table 2. The recall test consisted of a second single page containing a list of the 20 actions. Each action was preceded by a blank in which command names were to be written.

Materials for the performance test, in addition to the recall test described above, included three one-page documents illustrating the editing tasks they were to perform. The first of these documents required subjects to use only commands for pointer movement. The second document required the use of commands for deletion, which, by necessity, involved the use of commands for pointer movement. The final document illustrated more realistic editing tasks, requiring the use of all of the previously practiced commands plus text insertion. This will be refered to as the realistic editing task. All languages were implemented with an automatic insertion mode. Each document contained 10 editing tasks. The order in which the tasks were to be performed was clearly specified. These documents are presented in Appendix B.

During the performance task subjects worked at a Digital VT-100 terminal connected to a VAX 750 mainframe computer. An IBM XT microcomputer was used to collect keystroke data. Time was recorded with a hand held stop watch.

Procedure.

Subjects were tested individually. Each was randomly assigned to one of the three experimental languages. The languages were taken from those used in Experiment 1. Ten subjects were tested in each language. They were first given a brief oral introduction to text-editing. They were given the single page of 20 command names paired with their actions and were allowed 5 minutes to study the list. After the study period subjects were given the prompted recall form and recalled as many command names as possible. Subjects were allowed as much time as necessary to complete this task. Number of correctly recalled command names was recorded.

Subjects were then seated at a computer terminal and were again given the list of command names and their actions, which they were allowed to refer to at any time during the performance tasks. Subjects were provided one at a time with the three one-page documents illustrating the tasks they were to perform. They were not allowed any practice prior to each test. They completed all tasks on the movement document before being given the document illustrating deletion tasks. When the deletion tasks were completed, subjects where given the realistic editing document. Time required to complete each set of tasks was recorded. Each subject's edited document and record of individual keystrokes were saved for future error analysis.

Results and Conclusions

In the recall phase of Experiment 2, mean number of correctly recalled command names was 9.10 (SD = 3.98), 16.0 (SD = 4.35), and 19.9 (SD = 0.32) out of 20 for the anti-organized, organized and orthogonal languages, respectively. A one-way analysis of variance revealed a significant difference in the number of command names recalled [F(2,27) = 25.74, $p < .001$]. The least significant difference for the 0.05 level was calculated to be 2.64, revealing that all differences among the three means were reliable. These results replicate the basic findings of Green and Payne (1984) providing support for their hypothesis that the overall structure of a language is a more important influence on ease of learning than any individual command name features.

In the performance phase of Experiment 2, mean times necessary to complete each of the three editing tasks were calculated for the three language conditions. These data are presented in Table 3.

The experiment involved a 3 X 3 mixed design with experimental language as the between subjects factor and text-editing tasks as the within subjects factor. Analysis of variance yielded significant main effects of language [F(2,27) = 6.06, $p < 0.01$] and of task [F(2,54) = 27.25, $p < 0.001$] as well as an interaction [F(4,54) = 3.90, $p < 0.01$]. The least significant difference between language means at the 0.05 level was 183 seconds. Only the performance times of the subjects learning the orthogonal and anti-organized languages differed significantly, and then only on the pointer movement task. Although this was the only reliable difference in time necessary to complete the editing tasks, differences in the realistic task approached significance (difference of 174 seconds) and subjects learning the orthogonal language consistently required less time to complete the tasks than subjects learning either of the other languages.

Table 3: Results from experiment 2.

Mean times and standard deviations, in seconds, necessary for ten subjects in each of the experimental languages to complete the text-editing tasks.

	ORTHOGONAL	ORGANIZED	ANTI-ORGANIZED
MOVEMENT	259 (SD = 91.4)	452 (SD = 146.8)	622 (SD = 183.7)
DELETION	506 (SD = 138.3)	602 (SD = 214.4)	598 (SD = 130.6)
REALISTIC	564 (SD = 102.8)	675 (SD = 219.9)	738 (SD = 197.2)

These results provide some evidence that the additional keystroke required by the orthogonal language, in this experiment, did not slow performance.

To determine whether any relation exists between performance on the recall test and performance on the text-editing tasks, mean times for the combined tasks were calculated for each subject. Recall scores and mean time values were correlated, $r\,(27) = -0.60$, $p < 0.01$, indicating that pen-and-paper tests can be useful indicators of users' actual performance with command languages.

Error rates were negligible for subjects in each of the experimental language conditions, therefore no analysis was performed.

DISCUSSION

The present results provide support for Green and Payne's (1984) hypothesis that consistency between language rules is more important than the match between any one command and its name. An orthogonal language is easier to learn and to use than either a language with conflicting rules of organization or a language with consistent, but non-orthogonal organization. This is due to the fact that when each basic factor necessary for the execution of an action is independently represented in the command, any command, whether previously learned or unknown, can be derived from the point of intersection of the relevant factors.

Command Language Design

A designer who wishes to design a command language in which the commands are orthogonal to each other must first identify the basic factors that are necessary for the execution of the actions. Next, each of these basic factors would be assigned a unique set of symbols representing different values within the factor. For example, direction is a basic factor that can have different values relative to the position of the pointer. It is at this level of assigning symbols to the different values of a factor that mnemonics come into play. Finally, a consistent syntax should be used to combine the symbols.

The above steps, though simple, seem to have been overlooked by all researchers. Several have mentioned orthogonality as a means for increasing the internal organization of a language but have invented commands that used too few symbols to independently represent each basic factor of the action (Green and Payne, 1984; Perlman, 1984). As mentioned earlier, many text editing commands require the user to specify the mode of the operation, the unit of text the operation should be performed on, and the direction, relative to the pointer's current position, in which the operation should be performed. An orthogonal design requires three symbols under these conditions. Commands for actions in domains other than text-editing could be similarly broken down into their basic factors. These basic factors would then be represented independently of each other in the command language.

There are some situations in which an orthogonal design may be difficult to implement. For example, in text-editing, block manipulation actions cannot be executed with a single three-keystroke command. Therefore, to maintain orthogonality these actions must be divided into several subcommands. To move a block of text would require a cut command and a pointer movement command, each of which contains three basic factors, and a paste command to insert the text at the proper location. The paste command contains only one factor and therefore does not fit the previously specified othogonal design. To maintain simplicity, it may be necessary to have subsets of orthogonality in which commands with the same number of basic factors are grouped together.

A second problem is the necessity for multiple keystroke commands. Initially this increases ease of learning, but as the user becomes more familiar with the system, the additional keystrokes may become a nuisance. Advanced and expert users are no longer concerned with learning the system. They are concerned, however, that the language be concise and efficient. We suggest than any command language intended for expert, as well as novice users, contain default commands based on the larger commands but minimizing keystrokes. For example, if all commands were to default to the forward direction then the forward direction symbol could be made optional.

Another way to overcome the problem of additional keystrokes might be to allow chording. This would allow the user to issue a command by pressing a combination of keys simultaneously. Some common examples of chording are found in the performance of pianists and typists. There is some evidence that chording is not difficult to learn (Gopher, Koenig, Karis, and Donchin; in press).

Extensions to Programming Language Design

We believe that the basic concept of an orthogonal design, as advanced in this paper, can be generalized to the design of programming languages. The use of orthogonality is not new in the domain of programming languages. Ghezzi and Jazayeri (1982), cite the differences in orthogonality between the way Pascal and ALGOL 68 implement parameter passing. Pascal's lack of orthogonality leads to many restrictions including the fact that files cannot be passed by value, components of a packed data structure cannot be passed by reference, and functions can only return values of a restricted set of types; they cannot return arrays, records, sets or files. On the other hand, the extreme orthogonality of ALGOL 68 limits the readability of the program by allowing virtually anything to be combined with anything else. This leads to extremely complex procedures. Consider the following example from ALGOL 68 cited by Ghezzi and Jazayeri (p. 261):

```
(real x,y,; read ((x,y)); if x<y then a else b fi) :=
b + if a:= a+1; a > b then c:= c+1;  +b
                  else c:= c-1; a
    fi
```

In Ada parameter passing has been implemented orthogonally while maintaining simplicity. This program feature is divided into two basic factors; how the parameter is to be used (input, output, or both) and how it is to be passed (by reference or by value and/or result). Ada allows any combination of these factors. The user declares each parameter as in, out, or in out and the compiler chooses between pass by reference or pass by copying based on efficiency considerations (e.g. the number of words that the parameter occupies and its data type) (MacLennen, 1983). By leaving it up to the compiler to determine how the parameters are passed, the designers of Ada have reduced the complexity of this feature for users and have prevented them from choosing an illogical combination of the two features or one that is technically difficult to implement.

ALGOL 68's use of orthogonality has also led to the criticism that orthogonality adds to the complexity of languages by necessitating the addition of combinations of features that are not useful, or are technically difficult to implement (Hoare, 1973; MacLennen, 1983). We argue that it is more important to break a programming language down into related subsets of features, and then implement each of these subsets orthogonally, than to create useless combinations simply to maintain orthogonality across the entire language. This could be accomplished by isolating the basic factors within a subset and changing only the symbols identifying these factors in order to change the meaning of a statement.

ALGOL 68 is also criticized for its confounding of orthogonality with generality, thereby reducing the language's simplicity. For example, ALGOL 68's for-loop construct can take any number of forms, allowing a sequence of control variable values to be generated by any number of assignments, while statements, or if-then-else statements. This greatly reduces the readability and clarity of the code. In the orthogonal design we have proposed, the programmer would be allowed to make one-for-one substitutions within a factor, but would not be allowed to use an unrestricted number of symbols simultaneously. Orthogonality could be carried to its logical extreme by allowing the order of factor combination to be totally unrestricted. Again this would greatly reduce the readability and clarity of the code. We argue that a language should have a clear and consistent syntax for the combination of factors.

CONCLUSIONS

The results of the present studies suggest several major conclusions. First, findings in psychological research of a relation between the degree of organization and later memory performance can be extended into the domain of command language design. Second, orthogonality as a principle of design, can contribute positively to the user's ability to learn the command language. Third, there are general steps that should be taken when attempting to build a high degree of consistency into a command language. Finally, pen-and-paper tests of recall and predictability can be useful in predicting performance with a language on real-life tasks.

Jorgenson, Barnard, Hammond and Clark (1983) have shown that designers normally deal with commands individually, rather than designing the language as a whole. In this paper we have argued that a language that is easy to learn and use cannot be created simply by combining various mnemonic symbols. Mnemonics and organization are not mutually exclusive principles. The use of well chosen symbols without explicit consideration of the overall organization of the language cannot lead to a good language, however a language with consistent organization can take full advantage of well chosen symbols.

REFERENCES

Black, J.B. and Moran, T.P. (1982). Learning and remembering command names. *Proceedings of the Conference on Human Factors in Computer Systems*, 8-11. Gaithersburg, Maryland. March.

Bousfield, W. A. (1953). The occurrence of clustering in the recall of randomly arranged associates. *Journal of General Psychology, 49*, 229-240.

Bousfield, W. A., Cohen, B. H. and Whitmarsh, G. A. (1958). Associative clustering in the recall of words of different taxonomic frequencies of occurrence. *Psychological Reports, 4*, 39-44.

Bower, G. H. (1970). Organizational factors in memory. *Cognitive Psychology, 1*, 18-46.

Card, S. K. (1982). User perceptual mechanisms in the search of computer command menus. *Proceedings of the Conference on Human Factors in Computer Systems*, 1-7. Gaithersburg, Maryland. March.

Cohen, G. H. and Bousfield, W. A. (1956). The effects of a dual-level stimulus-word list on the occurrence of clustering in recall. *Journal of General psychology, 55*, 51-58.

Ghezzi, C. and Jazayeri, M. (1982). *Programming Language Concepts.* New York: John Wiley & Sons.

Gopher, D., Koenig, W., Karis, D., and Donchin, E. (in press) The representation of movement schemas in long-term memory: Lessons from acquistion of a transcription skill, I. *Acta Psychologia.*

Green, T. R. G. (1983). Learning big and little programming languages. In Wilkenson, A. C. (Ed.) *Classroom Computers and Cognitive Science.* New York: Academic Press.

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

Hemenway, K. (1982). Psychological issues in the use of icons in command menus. *Proceedings of the Conference on Human Factors in Computer Systems*, 20-23. Gaithersburg, Maryland. March.

Hoare, C. A. R. (1973). Hints on programming language design. In *SIGACT/SIGPLAN Symposium on Principles of Programming Languages*, October.

Horowitz, E. (1984). *Fundamentals of Programming Languages.* Rockville, Maryland: Computer Science Press.

Jenkins, J. J., Mink, W. D. and Russell, W. A. (1958). Associative clustering as a function of verbal association strength. *Psychological Reports, 4*, 127-136.

Jenkins, J. J. and Russell, W. A. (1952). Associative clustering during recall. *Journal of Abnormal Social Psychology, 47,* 818-821.

Jones, M. R. and Zamostny, K. P. (1975). Memory and rule structure in the prediction of serial patterns. *Journal of Experimental Psychology: Human Learning and Memory, 104,* 295-306.

Jorgensen, A. H., Barnard, P., Hammond, N., and Clark, I. (1983). Naming commands: An analysis of designers' naming behaviour. In T. R. G. Green, S. J. Payne, and G. C. Van Der Veer (Eds.) *The Psychology of Computer Use.* London: Academic Press.

Kotovsky, K. and Simon, H. A. (1973). Empirical tests of a theory of human acquisition of concepts for serial patterns. *Cognitive Psychology, 4,* 399-424.

Mandler, G. (1967). Organization and memory. In K. W. spence and J. T. Spence (Eds.), *The psychology of learning and motivation. 1.* New York: Academic Press.

MacLennan, B. J. (1983). *Principles of Programming Languages: Design, Evaluation, and Implementation.* New York: CBS College Publishing.

Meyers, G. (1978). *Advances in Computer Architecture.* New York: John Wiley.

Orstein, P. A., Trabasso, T., and Johnson-Laird, P. N. (1974). To organize is to remember: The effects of instructions to organize and to recall. *Journal of Experimental Psychology, 103,* 1014-1018.

Payne, S. J. and Green, T. R. G. (1983). The user's perception of the interaction language: a two-level model. In *Proceedings CHI'83 Human factors in Computing Systems,* 202-206. New York: Association for Computing Machinery.

Perlman, G. (1981). Two papers in cognitive engineering: The design of an interface to a programming system, and MENUNIX: A menu-based interface to UNIX (User Manual). Report No. 8105 La Jolla, California: Center for Human Information Processing, Uni

versity of California, San Diego.

Perlman, G. (1984). Natural artificial languages: Low-level processes. *International Journal of Man-Machine Studies, 20,* 373-419.

Reisner, P. (1981). Formal grammar and human factors design of an interactive graphics system. *IEEE Transactions on Software Engineering, 5,* 229-240.

Restle, F. (1970). Theory of serial pattern learning: Structural trees. *Psychological Review, 77,* 481-495.

Restle, F. and Brown, E. R. (1970). Serial pattern learning. *Journal of Experimental Psychology, 1,* 120-125.

Roberts, T.L. (1979) *Evaluation of Computer Text Editors,* Technical Report SSL-79-9, XEROX Palo Alto Research Center, p. 145, 148-149.

Rosenberg, J. (1982). Evaluating the suggestiveness of command names. *Proceedings of the Conference on Human Factors in Computer Systems,* 12-16. Gaithersburg, Maryland. March.

Sakado, J. M. (1956). Individual differences in correlation between clustering and recall of meaningful words. *Journal of General Psychology, 54,* 183-190.

Tanenbaum, A. (1984). *Structured Computer Organization.* Englewood Cliffs, New Jersey: Prentice-Hall.

Tulving, E. (1962). Subjective organization in free recall of "unrelated" words. *Psychological Review, 69,* 344-354.

Tulving, E. (1967). Organization and interference. Paper presented at AAAS meeting. New York, December.

APPENDIX A

Experimental Materials Used in Prediction Test, Experiment 1

We are interested in how people generate text editing commands. The following is a list of operations you might want to do when text editing. Look at the following two examples and fill in the blanks below.

ESC → C	move the pointer forward a character
DEL ← W	delete the word to the left of the pointer
__________	move the pointer forward a word
__________	move the pointer forward a sentence
__________	delete the character to the right of the pointer
__________	delete the document to the left of the pointer
__________	delete the line to the right of the pointer
__________	delete the word to the right of the pointer
__________	move the pointer backward a paragraph
__________	move the pointer forward a paragraph
__________	move the pointer backward a word
__________	move the pointer forward a line
__________	move the pointer backward a line
__________	move the pointer backward a character
__________	delete the character to the left of the pointer
__________	move the pointer backward a sentence

APPENDIX B

Experimental Materials Used in Learning Performance, Experiment 2

NOTE: The documents used in Experiment 2 were chosen from Roberts, T.L. (1979)*Evaluation of Computer Text Editors*, Technical Report SSL-79-9, XEROX Palo Alto Research Center, p. 145, 148-149.

Movement Tasks: Instructions*

Move the pointer to each of the numbered spots, in the order indicated.

1. Move forward one word
2. Move to the end of the line
3. Move to the end of the document (file)
4. Move to the beginning of the line
5. Move to the beginning of the paragraph
6. Move to the end of the sentence
7. Move backward one word
8. Move backward one character
9. Move to the beginning of the document (file)
10. Move to the end of the paragraph

* The numbers associated with these tasks correspond to those on the document. They can be considered a description of the task the subject is to perform.

Movement Tasks: Document

[9] Dear[1] Madam:

I am taking the liberty of writing this letter to you, the Editor-in-[2]
Chief, because I believe that you personally may be interested in my
services.

I am twenty-four years old, unmarried, a graduate of Columbia
University, School of Journalism, 1969, rating among the first five of a
class numbering one hundred. During my college years, I worked for
three summers in the Production Department of Rivers and Company,
assisting in various capacities and learning methods and techniques in
preparation for an editorial career. In my junior and senior years, I
was Assistant Editor of the college newspaper, The Spectator.[10]

After my college graduation, I worked three years for the Benson
Publishing Company, in the Assistant Editor's office. My work comprised
editing manuscripts of many different types, helping in the interviewing of
prospective authors, conferring with the Assistant Editor about the
acceptance of manuscripts, and doing considerable research and rewriting
on some of those that were accepted.

For the last two years, I have been Assistant Editor at Wesley House.
In that capacity, I have handled a great deal of the fiction that
the firm has published during the past year.
Here, again, I have worked with authors, including much consultation and
collaboration while they were writing their manuscripts. This procedrue
saved the firm considerable editorial expense after the manuscripts were
accepted for publication.

My relations in my present position are mutuallly pleasant, but I feel I can
use my ability to still better advantage. I believe that my services are
worth $200 a week.

[5]I should sincerely appreciate the courtesy of an interview at your [8]
[7]convenience.[6] In that event, I shall of course bring with me the best of
references.

Very truly yours,
[4] Kelly M. Starr [3]

Deletion Tasks: Instructions*

Delete each of the sections within the boxes, in the order indicated and from the position of the number

1. Delete one word
2. Delete a line
3. Delete one character
4. Delete a sentence
5. Delete one word
6. Delete one character
7. Delete a sentence
8. Delete one character
9. Delete a line
10. Delete a line

* The numbers associated with these tasks correspond to those on the document. They can be considered a description of the task the subject is to perform.

Deletion Tasks: Document

1 $\boxed{\text{Dear}}$ Sandy,

2 $\boxed{\text{I just can't get used to the fact that you're not in our old home town}}$
any more, nor in the office with me. How long is it now
since you left here? The calendar says it is eight months, and you
can't argue with the calendar, though I'm inclined to do just that.

Neither of us is a very good co$\overset{6}{\boxed{\text{r}}}$respondent but I think it is my turn to
write, and first of all I want to say, give me more news about yourself.
Is the new job out there on the Coast proving worth your having made the
move? Is the manager easy to get along with, and does he appreciate
your talents and ability? I've heard he is rather "hardboiled." Have you
found a good apartment? Last time you wrote, you were still looking. How
about recreation? Is there a good bowling club for you to join?7 $\boxed{\text{I know}}$
$\boxed{\text{you'd be lost without one.}}$

As for me, you'll be glad to hear that I am to be promoted next month--
Assistant Sales Manager, no less! You'll have to address me as "Ma'am" after
this. I'll get considerably more salary and that will be mo$\overset{8}{\boxed{\text{p}}}$st welcome,
$\boxed{\text{with the cost of living apparently going up}}$10 indefinitely.9 $\boxed{\text{Lou Mayer and I}}$
plan to spend our vacation together next summer at Lake Placid. He likes
the outdoors as much as I do.

Chris Turner in our office--you remember her--has at last become engaged to
Par Macy, the Personnel Manager. We all saw that coming--or perhaps she
kkdidn't. And, oh yes, the town has condemned the property at 12 Walnut
Street. About time, everybody says. It certainly was an eyesore.

Well, that's $\boxed{\text{about}}$5 all. $\boxed{\text{How am I doing?}}$4 $\overset{3}{\boxed{\text{P}}}$lease do at least as well
when you answer, and let that be soon, Sandy.

As always,
Lee

Realistic Tasks: Instructions*

1. Change "e" to "a"
2. Change "Ninth" to "Tenth"
3. Change "for" to "because"
4. Correct spelling of "old"
5. Insert "our"
6. Insert "many"
7. Delete "are the"
8. Delete "that"
9. Delete "'s"
10. Correct spelling of "opportunity"

* The numbers associated with these tasks correspond to those on the document. They can be considered a description of the task the subject is to perform.

Realistic Tasks: Document

4 November 1951

D$\overset{1}{\text{e}\boxed{\text{e}}}$r Customer:

This is our $\overset{2}{\boxed{\text{Tenth}}}$ Anniversary--but you are having the party!

After all, that's absolutely appropriate, $\overset{3}{\boxed{\text{for}}}$ it's you, and other good and loyal customers like you, who, by their generous and continual patronage, have made our mail-order business flourish, so that each of our anniversaries has been bigger and better.

All our patrons, $\overset{4}{\boxed{\text{odl}}}$ and new, can enjoy the party for the next two weeks. And it will really be a party--with a dozen great bargains for you and hundreds of $\underset{5}{\wedge}$ other friends.

The enclosed post cards, which can be used for your orders, give you an idea of what is in store for you.
Note, for instance, how we have slashed prices on men's fine handkerchiefs with corded borders and rolled edges; spun nylon socks; three-year-guaranteed stainless-steel cutlery; long-wearing auto-seat covers; and $\underset{6}{\wedge}$ other items.

Look over the enclosed cards now, and find out what $\overset{7}{\boxed{\text{are the}}}$ articles
$^{8}\boxed{\text{that}}$ you want and need. We may not be able to offer you these wonderfully low anniversary prices again--these phenomenal savings for you, your family, your home.

You have a week$\overset{9}{\boxed{\text{'s}}}$ trial, free, of any starred article you select.
Merchandise is prepaid to your door. If you're not entirely satisfied, or return the goods, and owe us nothing.
This $\underset{10}{\boxed{\text{oportutiny}}}$ may not occur again. Mail your order cards IMMEDIATELY!

Yours for anniversary savings,
Dana Eliot Hastings
President, Buy-Mail Corporation
Enclosures