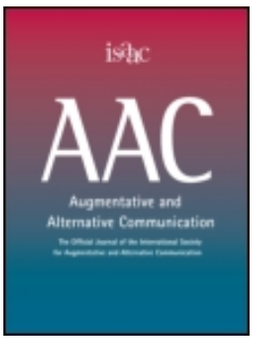

Augmentative and Alternative Communication

ISSN: 0743-4618 (Print) 1477-3848 (Online) Journal homepage: http://www.tandfonline.com/loi/iaac20

Modeling of user performance with computer access and augmentative communication systems for handicapped people

Heidi Horstmann & Simon Levine

To cite this article: Heidi Horstmann & Simon Levine (1990) Modeling of user performance with computer access and augmentative communication systems for handicapped people, Augmentative and Alternative Communication, 6:4, 231-241, DOI: 10.1080/07434619012331275514

To link to this article: https://doi.org/10.1080/07434619012331275514

0743-4618/90/0604-0231$3.00/0
AAC Augmentative and Alternative Communication

Modeling of User Performance with Computer Access and Augmentative Communication Systems for Handicapped People

Heidi M. Horstmann and Simon P. Levine
Rehabilitation Engineering Program, Department of Physical Medicine and Rehabilitation, University of Michigan, 1C335 University Hospital, Ann Arbor, Michigan 48109-0032, USA

The design of the user interface for an augmentative and alternative communication (AAC) or computer access system is a critical factor in determining a user's performance with a system. A comprehensive, quantitative, and accurate model for alternative access systems is needed to optimize both developers' design decisions and clinicians' system recommendations. This paper presents an application of one possible model, called the GOMS *(Goals, Operators, Methods, Selection Rules)* model (Card, Moran, & Newell, 1983). The model provides a comprehensive description of user behavior based on system-specific parameters as well as the cognitive, perceptual, and motor capabilities of the user. It can be used to predict both task execution and learning times, as well as points of excessive long or short term memory load. The GOMS model is applied here to three interfaces currently used in AAC and computer access systems in order describe and predict user performance, both qualitatively and quantitatively. The three interfaces are : (a) row-column letter scanning; (b) row-column letter scanning combined with word prediction after the first two letter selections only; and (c) row-column letter scanning combined with word prediction after each letter selection. Techniques for applying the GOMS model are discussed, as well as the results predicted by the model. Results for the three systems modeled here suggest the possibility that word prediction interfaces, developed as a faster alternative to row-column letter scanning, may actually be less efficient than letter scanning.

KEY WORDS: alternative access, computer access, computer access for handicapped people, user performance modeling

The personal computer has tremendous potential for improving the functional abilities of individuals with physical and cognitive disabilities. Some of this potential has already been realized, and many new educational, vocational, and recreational opportunities have opened up for individuals with disabilities through the use of the computer.

For a computer to be useful to individuals with disabilities, alternatives to the computer's hardware or software must often be developed. For example, a user who cannot physically use the standard keyboard must have an alternative means of accessing the computer, referred to as a computer access system. In addition, use of the computer as an augmentative and alternative communication aid for people who cannot speak requires a special user interface design, similar to that of a computer access system. Throughout this paper, both types of systems, that is, computer access systems with and without voice output, will be referred to as AAC (augmentative and alternative communication) systems.

The user interface of an AAC system provides the user with a way to interact with the system, both in entering new information into the system (user input) and receiving feedback from the system (system output). The quality of this interaction is a determining factor in user performance, so the design of the user interface is critical to the functionality of the system. Developers who make decisions on system design should have a consistent, quantitative means of ensuring the quality of their decisions. Similarly, clinicians who recommend systems for their clients should have a way to analyze the performance characteristics of an AAC system and not have to rely exclusively on clinical experience and short term user trials to make their recommendation.

For example, consider a system that allows the user to select text from both letter and word menus, where the items in the word menu change based on the letters selected. The questions presented by this system include: How many words should be in each menu? What is the effect of increased word list search time? Are there situations where it is inefficient to use the word menu? As a second example, considering an encoding system leads to questions such as: How does learning time increase as a function of number of encoded

items? What should be the criteria for a text segment to justify encoding (e.g., frequency of use, length of the text segment)?

One way to address these questions is to employ a mathematical model of an entire AAC system, including user interactions, hardware, and software. The goal of such a model is to accurately describe system behavior and user performance. If the model is accurate, it should be able to correctly simulate and predict the effect of a change in system parameters or user characteristics on performance. An advantage of a modeling approach is that it is usually much easier to simulate a proposed change using the model as opposed to determining its effect through user trials.

Model simulations are used extensively in fields like aerospace or automobile design, where it may be too expensive to create a physical prototype to test every new design concept. Similarly, in AAC and computer access, accurate and comprehensive modeling tools are needed, in order to estimate user performance in terms of both AAC system and user parameters. This will allow developers to make better design decisions and clinicians to better understand and predict the interaction between a user and a proposed system.

This paper explores the use of a modeling technique to represent user/AAC system behavior in order to provide both developers and clinicians with a framework that can improve the development and prescription of AAC systems. The first part of the paper discusses the rationale for modeling user behavior with AAC systems and reviews some of the previous work in this area. The second part describes the application of one modeling technique in particular, called the GOMS *(Goals, Operators, Methods, Selection Rules)* model (Card, Moran, & Newell, 1983). The GOMS model is used to describe user behavior and predict user performance for three scanning interfaces currently used in computer access and AAC systems.

The long-term goal of this work is to use the model to quantitatively predict AAC user/system performance and simulate a large range of user and system characteristics. This is not an easily achieved goal. However, the process of mathematically modeling the user/AAC system behavior offers, at a minimum, a valuable *qualitative* analysis, since it provides the opportunity to carefully analyze the interaction between the user and an AAC system, under a wide range of conditions. Therefore, the primary purpose of this initial effort is not to quantitatively compare the three interfaces chosen, but rather to demonstrate the usefulness of the GOMS modeling technique in understanding (and ultimately predicting) the effect of AAC system design on user performance.

Background

There are many AAC systems that are either commercially available or in the final stages of testing, with more packages being developed each year. These incorporate a wide range of physical input methods, such as expanded keyboards, head pointing devices, and breath-controlled switches. In addition, a variety of methods designed to enhance rate, such as symbolic encoding, abbreviation expansion, and word prediction, can be employed. In one sense, this proliferation of systems is advantageous, since these systems can offer individuals with disabilities a level of independence and achievement that might otherwise be unattainable.

However, there is a disadvantage to the continued development of additional systems; in the push to develop prototypes, important design factors may be ignored. Successful interaction with an AAC system involves both the physical and mental abilities of the user. The physical input technique used (e.g., single switch, expanded keyboard) determines the physical component of the user's performance, as measured by the text entry rate. The mental component of the user's performance includes all operations necessary to decide on a physical action, such as searching a word list for the desired word. Some users may have cognitive and/or perceptual impairments, which affect this mental component. These physical and mental factors must be explicitly addressed by both the designer and the clinician in order to produce and provide appropriate AAC systems.

However, specific analyses of these issues are rare, and those that do exist primarily focus on the physical efficiency of the system. For example, the number of switch activations per letter is one of the most common parameters analyzed (Goodenough-Trepagnier, Rosen, & Demsetz, 1982; Heckathorne, Voda, & Leibowitz, 1987). Another parameter which has been used for evaluating word predictive or abbreviation expansion interfaces is the ratio of the number of characters selected to the number of characters generated (Swiffin, Arnott, Pickering, & Newell, 1987; Vanderheiden & Kelso, 1987). While physical efficiency is an important design factor, it should not be separated from the parallel goal of mental efficiency, that is, minimizing the cognitive and perceptual load that the system imposes on the user. Certainly the idea that mental load can compromise user performance is not new (Soede & Foulds, 1986). However, these authors do not make explicit attempts to quantify the effects of mental load.

Other studies have presented models of cognitive and perceptual load for specific systems (Dabbagh & Damper, 1985; Gibler & Childress, 1982). Gibler and Childress (1982) discuss the role of cognitive time as a component of selection time but do not include it in their analysis due to the difficulty of measuring it. Dabbagh and Damper (1985) discuss the relative mental and physical components of selection time for their system but use only a gross empirical measure which integrates all of these components. There still remains a substantial need for generally applicable methods that provide an integrated analysis of the cognitive, perceptual, and motor aspects of user performance with AAC systems.

Methods

Review of GOMS Modeling

The GOMS model is particularly well-suited for modeling the user interface of a computer-based system. This model has been the focus of a great deal of research in the field of human-computer interaction and was chosen for several reasons. First, it is comprehensive, since it is based on a general theory of human information processing. Second, it provides quantitative results, which have been verified in several task domains. Finally, it is relatively simple and can therefore be applied "in the field" by a designer or clinician. It must be stressed that the validity of the GOMS model is not being tested by the results presented in this paper. Rather, we are illustrating the application of a model that has been validated in the field of computer-human interaction to the field of AAC.

The GOMS model was developed by Card, Moran, and Newell (1983), and refined by Polson and Kieras (1985), among others (Ziegler, Hoppe, & Fahnrich, 1986). The model is based on a general theory of human information processing, called the Model Human Processor. According to this theory, human activity is governed by three processors in the brain, the Cognitive, Perceptual, and Motor Processors. The Perceptual Processor takes sensory information and stores it for use by the Cognitive Processor. The Cognitive Processor then uses this information, as well as previously stored information in long-term memory, to decide how to respond. The role of the Motor Processor is to carry out the response that has been determined by the Cognitive Processor. Each of these Processors is characterized by the speed at which it performs its tasks; this speed is known as the cycle time.

A basic principle of the GOMS model is that within the context of a structured task, users act predictably to achieve their goals. The user's behavior can then be represented by a sequence of elementary steps (called Operators) defined by the *Goals* of the user and the constraints of the task. These *Operators* are packaged into coherent subtasks called *Methods,* which are analogous to subroutines in a computer program. Each *Method* is specified by a sequence of *Operators,* each of which represents some combination of cognitive, perceptual, or motoric actions by the user. When more than one *Method* exists for a given subtask, a construct called a *Selection Rule* is used to select one of the *Methods.* The final model is a list of statements that represent the *Goals, Methods, Operators,* and *Selection Rules* to provide a complete model of the user's behavior in pursuit of the overall goal, specifying each required step in the proper sequence.

The resulting model of the user's behavior is useful in a qualitative sense because it provides a complete picture of the user's interaction with the modeled system. Because each required user action can be specified, actions that are potentially difficult, time-consuming, or unnecessary can be identified clearly.

The GOMS model can also be used to make quantitative predictions about a user's performance with an AAC system. Several components of user performance, including task execution time and learning time, as well as points of excessively long or short-term memory load, can be predicted in terms of system and user parameters. These predictions can then be used during the design process to estimate the consequences of particular design decisions, or to compare a proposed design to alternative systems. Several studies, most of which use text editing as the paradigmatic task, have demonstrated that the GOMS model provides a good description of user behavior and predicts task execution time and learning time with reasonable accuracy (Card, et al., 1983; Polson & Kieras, 1985; Ziegler, et al., 1986). The rules for making these predictions are described below.

Estimation of Task Execution Time. The first step in predicting overall task execution time is to identify all possible ways in which the task can be achieved, represented by paths through the GOMS model. For example, with some AAC systems, text can be entered either by selecting individual letters or by selecting complete words, so there are two paths for the task of text generation. Each path is defined by statements in the model that are executed when the user follows the path. The execution time for a given path can be estimated by adding up the times required to execute each individual statement (Card, et al., 1983; Kieras, 1988). The individual statement times are estimated as follows: one cognitive cycle time per statement, plus any additional time required for statement execution, (e.g., key-hit time, visual search time, decision-making time). The additional statement time components are estimated by the GOMS analyst. For statements in which an action of some kind is performed, an *Operator* representing the action is defined by the analyst. For example, if a statement specifies that the user is required to press a switch, a "switch-hit" *Operator* can be defined to represent that user action. These Operators can be subdivided into Cognitive, Perceptual, and Motor Processor times to yield an estimate for the *Operator* time. For open-ended "mental" *Operators,* such as thinking of the next word to be typed, it may not be possible to accurately subdivide the *Operator* into the relative number of Cognitive, Perceptual, and Motor Processor times. A generic mental *Operator* time is used in this case (Card, et al., 1983). The overall task execution time estimate is the weighted average of individual path times, based on the probabilities of individual path execution during general system use. In the case of the systems modeled here, the individual paths are the different methods used to select letters or words, and the overall task execution time is the text generation rate.

Estimation of Learning Time. An empirical formula is used to estimate learning time with the GOMS model. It is the sum of 30 minutes for baseline learning time, 30 seconds for each statement in the model, plus any

additional memorization time, as described below (Kieras, 1988). Consistency in the design can reduce learning time. If two or more statements describe very similar or identical operations, they need only be counted once toward estimating learning time. The rationale behind this is that after learning one of two similar operations, the second one can be learned in roughly no time at all (Polson & Kieras, 1985). For example, if both letter and word selection involve hitting a switch when the selection is highlighted, then once the user has learned letter selection, learning word selection takes essentially no additional time. The final factor affecting learning time is long-term memory storage. Each piece of information retrieved from long-term memory must first be memorized by the user. The required memorization time depends on the number of "chunks" that are used to store each piece of information. A chunk is a set of one or more related items. For example, the letter "b" out of context takes up 1 chunk, and the word "cat" also takes up 1 chunk since the three letters form one coherent concept (Card, et al., 1983). Kieras (1988) uses 10 seconds/chunk for estimating memorization time. Chunks are summed by counting one chunk for each pattern in retrieval cue, one chunk for each pattern in retrieved information, and one chunk for the association between the retrieval cue and the information (Kieras, 1988).

Short-Term Memory Storage Requirements. Short-term memory storage is involved any time there is information gained in one statement and used in a subsequent statement. In the systems to be analyzed here, short-term memory is used to temporarily store the label and/or location of items to be selected. For example, if the user wants to select the letter "T", he/she must retain that letter in short-term memory for the duration of scanning until the T can be selected. The GOMS model analysis provides a means of estimating the number of information chunks in short-term memory at any given time as well as the storage time between retention and retrieval for each chunk. The number of statements that must be executed between retention and retrieval yields an estimate of the necessary storage time for that information (Kieras, 1988). This provides an estimate for the user's short-term memory capabilities required for system operation.

Alternative Input Systems Modeled

The GOMS model is used here to explore the performance characteristics of three AAC interfaces, each of which is designed for use by a user with severe disabilities who can activate only one or two switches. Each interface is modeled using the GOMS model, and the resulting performance predictions are used as a basis of comparison. Each interface is described below.

The "standard" row-column scanning interface consists of a letter matrix that is scanned automatically to allow the user to make a selection using a single switch. The user waits for the system to highlight a particular row, then hits the switch to select the row. The system then highlights successive letters in that row, until the user hits the switch again to select the desired letter. The letters are arranged in order of overall frequency of occurrence, as shown in Figure 1, so that the letters with the highest frequency of use require the fewest number of scan steps for selection (Foulds, Baletsa, & Crochetiere, 1975). This arrangement stays fixed, which simplifies user memorization of letter position. Text is generated by selecting each letter from the letter matrix one by one.

The other two interfaces modeled add word prediction to simple letter scanning in an attempt to improve user performance. These systems exploit the redundancy of the English language in order to predict the user's desired word, thereby reducing the number of physical actions required of the user (Gibler & Childress, 1982). It is assumed that the predictive interfaces use the same letter matrix arrangement described above.

The first predictive interface studied is a variation on the PACA system, developed at Northwestern University (Heckathome & Leibowitz, 1985), referred to here as PACA_2 to distinguish it from the actual PACA interface. The first two letters of every word are selected using standard single-switch row-column scanning. When the second letter is selected, the letter matrix is replaced by a list of the seven most likely words that start with the two selected letters, as well as a "Next" command at the top of the list. Selections are made from the prediction list using one-dimensional single switch scanning. When a word is selected from the list, the word list is replaced by the letter matrix for the start of the next word. If the desired word is not in the first prediction list, the user can select "Next" to see a second prediction list. A second "Next" selection brings the letter matrix up again, and the word must be completed by selecting each of the remaining letters.

The second predictive interface analyzed is a scanning version of the PAL system, developed at the University of Dundee, Scotland (Arnott, Pickering, Swiffin, & Battison, 1984). The major differences between it and the PACA_2 system are (a) the word list and letter matrix are on the screen at the same time, and (b) predictions are made even before a letter is selected and are refined as subsequent letters are selected. Before each selection, the 10-word prediction list can

sp	E	A	R	D	U	V
T	O	I	L	G	K	
N	S	F	Y	X		
H	C	P	J			
M	W	Q				
B	Z					

Figure 1. Arrangement of letters in the standard row-column scanning system. Letters are arranged in order of frequency of occurrence, with the upper left corner as the most frequently chosen character.

be searched for the desired word. If it is there, the user hits a switch to initiate one-dimensional scanning of the word list. If not, the user hits a second switch to initiate row-column scanning of the letter matrix.

GOMS Models for the Three Interfaces

A GOMS model was constructed for each of the three scanning interfaces, following standard GOMS practice used in the field of human-computer interaction (Kieras, 1988). Additional details regarding the methods for modeling the interfaces and the resultant GOMS models for them are presented in Appendix A. One outcome of this modeling is a set of equations which can be used to predict text generation rates for each interface. These equations are based on a number of input parameters such as the user's basic Processor times (cognitive, perceptual, and motor), system parameters (e.g., scan rate, probability of desired word appearing in a word list, etc.), and *Operators* specific to the operation of the modeled system (e.g., switch hit, search of a word list, etc.). Estimates of learning time and short term memory load are also produced from the modeling process.

Model Simulations

Model simulations are performed by varying values for model input parameters and comparing the resulting outputs. The first step in comparing predicted text generation rates is to establish a set of nominal parameter values for all of the model input parameters used in the equations for each system. A list of these input parameters and the nominal values used in the simulation trials can be found in Appendix B. A rationale for the nominal values chosen is also presented there. A brief qualitative description of the model input parameters is provided below.

First, the basic Cognitive, Perceptual, and Motor Processor cycle times are parameters as defined by the Model Human Processor discussed above. Second, system characteristics are represented by four parameters: the system scan rate for the average number of letters/word, the average number of scans required for a word menu selection, and the probabilities of a word appearing on a particular word menu. Finally, several parameters specific to the user are also required. Some of these represent concrete actions, such as time to hit a switch or search a word menu. Others represent decisions that the user must make while using a particular system, according to the GOMS model constructed for that system. For example, for the PACA_2 system, the GOMS model specifies that the user must decide if the next selection represents one of the first two letters of a word, in order to choose the correct *Method* for item selection. Further detail is supplied in Appendix B.

Results

Task Execution Time

The results of simulation trials to predict overall text generation rate using the nominal parameter values are shown in Figure 2. All three predicted rates are slow, with the highest being the standard row-column (or "R/C") system at 3.58 words/minute (wpm). For the word prediction interfaces, the PAL system rate is somewhat higher at 3.16 wpm than the PACA_2 system at 2.92 wpm.

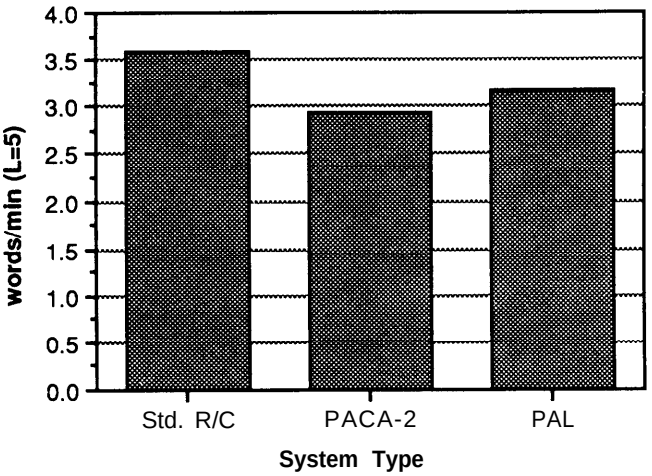

Figure 2. Predicted text generation rates for each interface. All parameters have nominal values.

This simulation trial predicts that the standard R/C scanning system is faster than the predictive interfaces. However, the nominal rate estimates are dependent on several imprecise parameter estimates, particularly in the predictive interface parameters. The next step, then, is to determine the sensitivity of the rate prediction to changes in the nominal parameter estimates.

Dependence on Number of Letters/Word

An increase in the average number of letters/word will decrease the text generation rate in words/minute regardless of the system type. Figure 3 shows the predicted text generation rate for each system plotted against the number of letters/word (L) when it is varied from 4.5 to 6 while all other parameters are kept at nominal values. It can be seen that the standard R/C system is much more sensitive to changes in L than either of the predictive interfaces. This is because the standard R/C system has only one selection path, single letter selection, so the number of letters/word is the same as the number of selections executed. With the predictive interfaces, a change in L affects only those selection paths in which the final letters of a word are selected, and this occurs only if the word is not successfully predicted, or roughly 30% of the time. Therefore, even though the predicted text generation rate for the standard R/C system remains higher than that of the other two systems throughout the range of L, the difference between the rates for the standard R/C and word predictive interfaces decreases as L increases.

Dependence on PACA_2 Prediction Parameters

As discussed in Appendix B, the two parameters specific to the PACA_2 system are w_1 and w_2, the probabilities that a word is on the first or second word

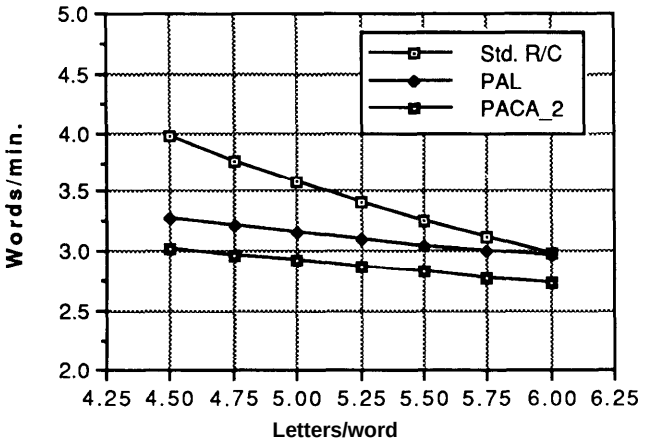

Figure 3. Predicted text generation rates for each interface as a function of average word length. All other parameters have nominal values.

lists, respectively, given a chance, *p,* that the word is in the dictionary. It is difficult to make confident estimates for system prediction parameters without knowledge of exact dictionary contents. Therefore, it is important to analyze how sensitive the overall rate prediction is to the values of these parameters. Figure 4 shows the predicted text generation rate for the PACA_2 system compared to standard R/C scanning, as w_1 is varied from 0.5 to 1.0, with p held at 0.7. Even with $w_1 = 1.0$, indicating that all words in the dictionary are presented on the first word list, the PACA_2 rate is predicted to be over 0.5 wpm slower than that of standard R/C scanning. In addition, the rate increases very slowly, from 2.86 wpm to 3.01 wpm over the range of w_1, which demonstrates that the rate equation is fairly insensitive to the values for w_1 and w_2. The effect of varying the proportion of words present in the dictionary, *p,* was not studied in these initial simulation trials.

Dependence on PAL Prediction Parameters

The sensitivity of predicted text generation rate to changes in two of the PAL prediction parameters was also studied. The parameters are defined as x_1 and x_2, the probabilities of a newly successful word prediction following selection of the first (x_1) or second (x_2) letters of the word. Figure 5 shows the predicted text generation rate for the PAL system compared to standard R/C scanning, as the prediction parameters are varied together, with $x_1 = x_2$ over the range of 0.25 to 0.40. This range was chosen because the parameters must be at least as large as the nominal estimate for x_0 (the probability of word prediction before any letters have been selected), while allowing x_3 (the probability of word prediction after the third letter selection) to remain positive (see Appendix B). The graph shows an estimated rate increase of 0.09 wpm for each increase of 0.05 in the prediction probabilities, with a maximum rate of 3.22 wpm at $x_1 = x_2 = 0.40$. This maximum rate is still 0.36 wpm below the estimated rate for standard R/C scanning.

Learning Time Requirements

Standard R/C Scanning. The GOMS model for the standard row-column scanning interface contains only seven statements, each of which takes an estimated 30 seconds to learn. Therefore, the base learning time can be estimated as: 30 min. + (30 sec) (7 statements) = 33.5 min. There are no consistency gains, but long-term memory storage time must be included. The user must memorize the location of all 27 characters in the letter selection menu in order to avoid excessive visual search time. The retrieval cue for this storage consists of "location-of letter-name," which takes two chunks. The retrieved information is a (row, column) coordinate, which also takes two chunks. Adding the association

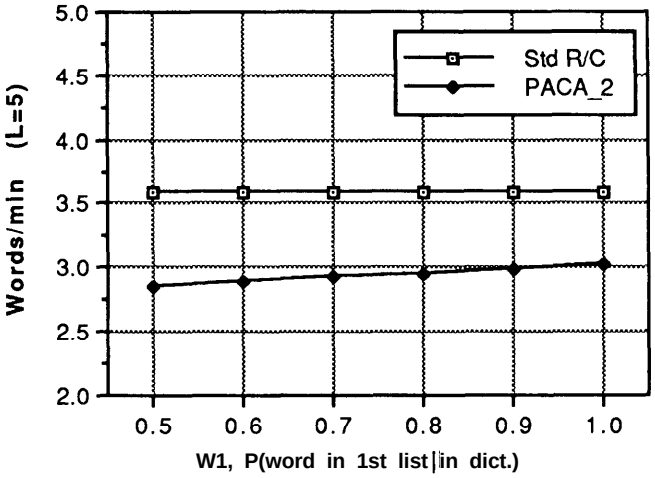

Figure 4. Predicted text generation rate for the PACA_2 interface as a function of w_1, the probability of a word appearing on the first word list, given that it is in the dictionary. The probability of a word being in the dictionary is fixed at 0.70, and all other parameters have nominal values. Predicted text generation rate for the standard row-column scanning interface is also shown for comparison.

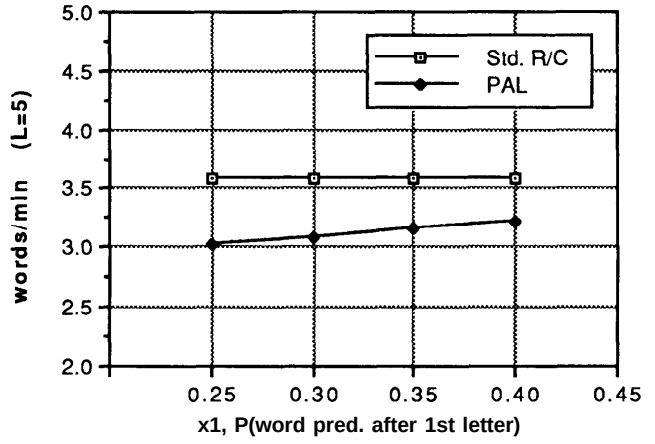

Figure 5. Predicted text generation rate for the PAL interface as a function of x_1 and x_2, the probabilities of successful word prediction following the first and second letter selections, respectively; x_1 and x_2 are varied together, with $x_1 = x_2$. The probability of a word being in the dictionary is fixed at 0.75, and all other parameters have nominal values. Predicted text generation rate for the standard row-column scanning interface is also shown for comparison.

chunk, five chunks are required for each letter location. At 10 set/chunk, the total time required to memorize the 27 letter locations can be estimated at 1350 seconds. The overall learning estimate, then, is: 33.5 min. + 1350 sec. = 56 min.

The PACA_2 System. The GOMS model for the PACA_2 system contains 29 statements, each of which takes an estimated 30 seconds to learn. Therefore, the base learning time can be estimated as: 30 min. + (30 set) (29 statements) = 44.5 min. Small learning transfer gains exist, which can be subtracted from the base learning time. The *Methods* for choosing a letter and choosing a word are essentially the same. Therefore, the three statements in the word selection *Method* can be subtracted from the total number of statements, yielding a learning time savings of 90 seconds.

To complete the learning time estimate, long-term memory storage time must be included. Memorization of the letter matrix adds 1350 seconds to the estimated learning time. The user must also memorize the name of the command for moving to the next word list and its location in the selection menus. The time required to memorize these eight chunks of information can be estimated at 80 seconds. The overall learning time is then: 44.5 min. – 90 sec. + 1430 sec. = 66.8 min.

The PAL System. The GOMS model for the PAL system contains 33 statements, each of which takes an estimated 30 seconds to learn. Therefore, the base learning time can be estimated as: 30 min. + (30 sec) (33 statements) = 46.5 min. Small learning transfer gains exist because the *Methods* for choosing letters and words are very similar, so two statements can be eliminated from the learning time calculation. Additional long-term memorization time is required only for the 27 character locations, which adds 1350 seconds to the total learning time. The total estimated learning time is therefore 68 minutes.

Figure 6 shows the overall estimated learning time for all three systems, with the relative contribution of each component: base learning time, time to learn task steps, and long-term memory time. The base learning time for each system is 30 minutes, as discussed above. The long-term memorization times for each system are also basically the same. The major component of long-term memory time is the memorization of the 27 letter matrix positions which takes 22.5 minutes. The standard row-column scanning interface has the shortest task step learning time, due to the small number of statements in its GOMS model; it is therefore predicted to have the shortest overall learning time.

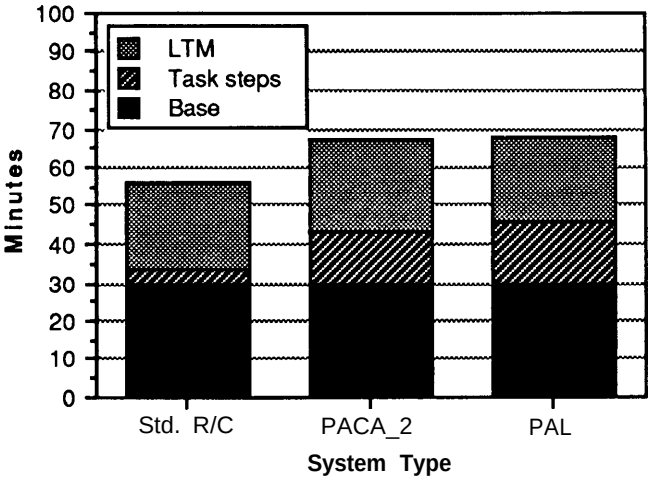

Figure 6. Predicted learning times for each interface. The three components of learning time are shown: base learning time, task steps, and long-term memory time.

Short- Term Memory Requirements

None of the systems modeled here places excess demands on short-term memory capacity or retention time. The largest amount of storage required at any one time is three chunks, which is safely below the five-chunk limit suggested by Kieras (1988) and all required retention times are less than 1 second.

Discussion

The specific GOMS models used in these simulation trials have not been validated by comparison of predicted results with actual user trials. Thus, the quantitative results obtained must be considered as preliminary. Nonetheless, there are several reasons to seriously consider these preliminary results. First, the GOMS modeling method has been validated for very similar applications (e.g., text editing). Second, detailed task analysis provided through GOMS modeling provides a good qualitative rationale for the results obtained. Finally, similar modeling approaches were used for all three AAC systems studied, which suggests that the quantitative results may have validity for relative, if not absolute, comparisons. With these considerations in mind, the following discussion of the results is presented.

Task Execution Time

Task execution time refers to the time it takes to perform the overall task. In the case of an AAC or computer access system, the overall task is to generate text to be spoken in a conversation, printed, or used as input to an application program. The ideal case is for the disabled user to approach rates achieved by able-bodied individuals, typically 35 to 40 wpm for typing and 100 to 200 wpm for speaking. These are unrealistic for a single or dual switch scanning system. The minimum acceptable rate should be above 3 wpm because at rates below this point, conversation breaks down due primarily to the receiver's impatience (Good-enough-Trepagnier, Galdieri, Rosen, & Baker, 1984). Goodenough-Trepagnier et al. (1984) have shown that receivers' impatience decreases markedly at a rate of 5 wpm, which makes this rate a reasonable target for a minimally acceptable rate. The preceding GOMS analysis provides estimates of task execution time for each system under a variety of conditions. The surpris-

ing overall result of this analysis is that none of the three interfaces approaches 5 wpm using nominal parameter values, even though user parameters correspond to those of an able-bodied user. In addition, the two predictive interfaces are at a consistently slower rate than the standard R/C system, with the PAL system somewhat faster than PACA_2, under almost all conditions.

There are several factors that contribute to the predicted slowness of the PACA_2 and PAL predictive interfaces. First, the number of statements to be executed in use of the predictive interfaces is much greater than the mere seven statements used in the standard R/C system; this reflects the relative complexity of the predictive systems. Second, use of the predictive systems requires additional mental *Operators,* such as visual search time and word-found matching, which increase the overall text generation rate. Third, the degree of word prediction success, which is limited by the size of the dictionary, also affects the estimated rate for the PACA_2 and PAL systems. An overall word prediction success of 70 to 75% may not provide enough word selection opportunities to counteract the mental overhead involved in using the more complicated predictive interfaces. In addition, when a word *is* selected, the length of the average word is too short to provide sufficient keystroke savings.

The proportion of words present in the dictionary is crucial to text generation speed. It should be noted that a major feature of both the PACA_2 and PAL systems is that the dictionary contents change dynamically based on the user's word usage. This feature may significantly increase the proportion of words present in the dictionary over time with a resulting increase in text generation rate. This is a user-specific system feature which cannot be easily modeled with the GOMS model. However, the GOMS model can be used to develop criteria for the proportion of words needed to be found in the dictionary in order to achieve a predefined performance level.

Finally, certain characteristics of the GOMS model itself may affect the predictions. The GOMS model as applied here is a serial model; it assumes that steps are executed in series rather than simultaneously. This restriction may lower the predicted task execution times, since in reality, the user may be able to execute two or more steps at once. For example, a user may be able to perform a cognitive activity, like placing the next pending selection in short-term memory, while completing a motoric activity, like hitting the selection switch. Recent enhancements to the GOMS model have made it possible to model such overlapping processes and may lead to more refined predictions of performance time (John, 1988).

In addition, the GOMS models used here do not account for errors in task execution, which certainly affect task execution times. The model can be used, however, to analyze methods for error recovery, in order to predict which *Methods* are most efficient when errors do occur.

Learning

System learning time should be as short as possible, since systems that are difficult to learn will be less acceptable to the target user. Rubinstein and Hersh (1984) propose a "10 minute rule" as a criteria for learning the basics of a system. This may be impossible to achieve as some published estimators of learning time use a base learning time of at least 30 minutes (Kieras, 1988; Polson & Kieras, 1985). A more reasonable goal for learning time may be 10 minutes in addition to base learning time, for a total of 40 minutes.

None of the estimates for the three modeled interfaces meets this threshold of 40 minutes. Note, however, that the estimated learning times include 22.5 minutes for memorization of the 27 letter matrix positions. Memorization of these positions is not essential for use of any system. Therefore, the time required for this memorization can be subtracted from the estimated learning time to give an minimum learning time estimate. When this is done, the learning time prediction for standard R/C system is well under the 40-minute threshold, and those for the two predictive interfaces are only a few minutes greater than 40 minutes.

The accuracy of estimating absolute learning time with the GOMS model has not been well validated in previous studies. Therefore, it may be more prudent to view the current learning time estimates as providing comparisons between systems, rather than absolute learning time predictions. When this is done, the results are as expected; the standard row-column scanning interface is predicted to have shorter learning time than the more complicated word prediction interfaces.

Conclusion

This research represents an initial stage in the development of a model that has the potential to become an extremely important tool in the design and prescription of computer access and communication aids for people with disabilities. The results presented here raise the question as to whether word prediction interfaces, developed as a faster alternative to row-column letter scanning, are actually less efficient than the letter-only interface. The model also provides insight into possible reasons for this surprising result.

The quantitative validity of these results is dependent upon the accuracy of the GOMS model descriptions and input parameters. Therefore, one direction for future research is to study the behavior of actual users to determine the validity of the GOMS model predictions. It may be that the current GOMS models do not provide accurate quantitative predictions in an absolute sense, given the limitations of the models discussed above. However, these limitations may not prevent the model from making accurate predictions of the relative performances among different interfaces or techniques. Indeed, if previous validation of the GOMS approach for analysis of text editing is assumed to carry over to the present application, then further sensitivity analysis

of input parameters can be expected to yield at the very least qualitative information about the value of one approach over another. Modeling an entire system is a time consuming task. However, by modeling various paradigms common to many different systems, criteria can be developed for system optimization, (e.g., determine the number of items to be presented in a menu or the efficacy of a linear vs. binary search strategy). The previous discussion on percentage of words to be found in a dictionary is a good example for potential application of this model.

It should be noted that optimizing task execution time, learning time, and short-term memory load of a system does not necessarily guarantee user acceptance. For example, some users may prefer word prediction systems simply because it helps their spelling. This may actually improve their performance over simple letter scanning because it reduces their error rate. Or they may perceive their performance to be better because they are less concerned with making errors. Finally, word completion may significantly enhance the quality of output for a semiliterate user, even if sheer number of letters/min is lower than single letter scanning. This example provides an important reminder of model limitations and the need to integrate use of the model with clinical skill.

Modeling of computer access and AAC systems has many potential benefits. With the proper model, a developer or clinician can create a qualitative analysis that provides an illuminating picture of a user's interaction with the system, and the possibility even exists for the creation of accurate *quantitative* descriptions. The GOMS model appears to be particularly well suited to this challenge, because it offers a comprehensive treatment of the cognitive, perceptual, and motor aspects of system use and provides both qualitative and quantitative levels of analysis.

Acknowledgments

This paper has been presented in part for presentation at the Eighth Annual Conference of the Cognitive Science Society held in August, 1989 and is based on a directed study project, entitled "User Performance Modeling for Computer Access and Augmentative Communication", written by the first author in partial fulfillment of the requirements for an M.S. degree in Bioengineering at the University of Michigan. The primary advisor for this project was Simon P. Levine, Ph.D.

Address reprint requests to: Simon P. Levine, Ph.D., Director, Rehabilitation Engineering Program, University of Michigan, 1 C 335 University Hospital, Ann Arbor, MI 48109-0032, USA.

REFERENCES

Arnott, J. L., Pickering, J. A., Swiffin, A. L., & Battison, M. (1984). An adaptive and predictive communication aid for the disabled exploits the redundancy in natural language. *Proceedings of the Second International Conference on Rehabilitation Engineering* (pp 349-350). Ottawa, Canada.

Card, S., Moran, T., & Newell, A. (1983). *The Psychology of Human-Computer Interaction.* Hillsdale, NJ: Erlbaum Associates.

Dabbagh, H. H. & Damper, R. I. (1985). Average selection length and time as predictors of communication rate. *Proceedings of the Eighth Annual Conference on Rehabilitation Technology* (pp. 404-406). Memphis, TN.

Foulds, R., Baletsa, G., & Crochetiere, W. (1975). The effectiveness of language redundancy in non-verbal communication, *Proceedings of the Conference on Devices and Systems for the Disabled* (pp. 82-86). Philadelphia.

Gibler, C. D. & Childress, D. S. (1982). Language anticipation with a computer based scanning communication aid. *Proceedings of the IEEE Computer Society Workshop on Computing to Aid the Handicapped* (pp 11-15). Charlottesville, VA.

Goodenough-Trepagnier, C., Galdieri, B., Rosen, M. J., & Baker, E. (1984). Slow message production rate and receivers' impatience. *Proceedings of the Second International Conference on Rehabilitation Engineering* (pp 347-348). Ottawa, Canada.

Goodenough-Trepagnier, C., Rosen, M. J., & Demsetz, L. (1982). Determinants of rate in communication aids for the non-vocal motor handicapped. *Proceedings of the Human Factors Society—26th Annual Meeting* (pp 172-175).

Heckathome, C. W. & Leibowitz, L. J. (1985). PACA: Portable anticipatory communication aid. *Proceedings of Eighth Annual Conference on Rehabilitation Technology* (pp 329-331). Memphis, TN.

Heckathome, C. W., Voda, J. A., & Leibowitz, L. J. (1987). Design rationale and evaluation of the portable anticipatory communication aid—PACA. *Augmentative and Alternative Communication, 3,* 170-180.

Horstmann, H. M. (1987). *User Performance Modeling for Computer Access and Augmentative Communication.* Directed Study Project, Bioengineering Department, University of Michigan, Ann Arbor.

John, B. E. (1988). *Contributions to Engineering Models of Human-Computer Interaction.* Ph.D. dissertation, Department of Psychology, Carnegie Mellon University, Pittsburgh, PA.

Jones, L., & Wepman, J. (1966). A *Spoken Word Count.* Chicago: Language Research Associates.

Kieras, D. E. (1988). Towards a practical GOMS model methodology for user interface design. In M. Helander (Ed.), *Handbook of Human-Computer Interaction.* Amsterdam: Elsevier Science Publishers (North-Holland).

Kucera, H., & Francis, W. N. (1967). *Computational Analysis of Present Day American English.* Providence, RI: Brown University Press.

Polson, P. G., & Kieras, D. E. (1985). A quantitative model of the learning and performance of text editing knowledge. *CHI '85 Proceedings* (pp 207-212). San Francisco, CA.

Rubinstein, R., & Hersh, H. (1984). *The Human Factor.* Bedford, MA: Digital Press.

Soede, M. & Foulds, R. A. (1986). Dilemma of prediction in communication aids and mental load. *Proceedings* of *Ninth Annual Conference on Rehabilitation Technology* (pp 357-359). Minneapolis, MN.

Swiffin, A. L., Arnott, J. L., Pickering, J. A., & Newell, A. F. (1987). Adaptive and predictive techniques in a communication prosthesis. *Augmentative and Alternative Communication, 3,* 181-191

Vanderheiden, G. C., & Kelso, D. P. (1987). Comparative analysis of fixed-vocabulary communication acceleration techniques. *Augmentative and Alternative Communication, 3,* 196-206.

Ziegler, J. E., Hoppe, H. U., & Fahnrich, K. P. (1986). Learning and transfer for text and graphics editing with a direct manipulation interface. *CHI '86 Proceedings* (pp 72-77). Boston, MA.

APPENDIX A

GOMS MODELS FOR THE THREE INTERFACES

Techniques for generating performance prediction equations using the GOMS model are described below for each interface. The case of standard row-column scanning is discussed in detail; similar principles apply to the other two interfaces.

Standard Row-Column Scanning. The GOMS model that was constructed for the standard row-column scanning interface contains seven statements, as shown below. The statements are written in a formal GOMS modeling language called NGOMSL. The complete methodology is described elsewhere (Kieras, 1988) but a general explanation is provided below. Each statement corresponds to either a *Goal, Operator, Method,* or *Selection Rule.* For example, statement (1.2) states that the subgoal of choosing an individual letter must be accomplished in order to achieve the overall *Goal* of entering a body of text. To initiate the subgoal, statement (1.2) transfers control to statement (2.1) which defines the *Method* required for achieving the letter choosing subgoal. Each statement in the letter choosing *Method* is executed in sequence until the *Method* has been completed, at which time statement (2.3) reports success, and control is returned to statement (1.3). The user must then decide whether to continue the process, based on whether all letters in the desired text have been selected.

It should be noted that statement (2.2), the row-column scan Operator, can itself be broken down into a number of component steps, such as visual search and switch hit. Clearly, an understanding of the component steps is necessary, and some analysts may choose to include them explicitly in the GOMS model. However, in this case, a parametric description of row-column scanning in terms of its component steps had previously been developed, using GOMS techniques (Horstmann, 1987). Therefore, we have chosen to use the results of that work to simplify our current GOMS model.

(1.1) *Method* to accomplish *Goal* of entering text
(1.2) Step 1. Accomplish *Goal* of choosing a letter
(1.3) Step 2. *Operator* to Decide: if text-complete, then report *Goal* accomplished.
(1.4) Go to Step 1.
(2.1) *Method* to accomplish *Goal* of choosing letter
(2.2) Step 1. *Operator* for row-column scan to select letter
(2.3) Step 2. Report *Goal* accomplished.

The only possible selection path using this interface is a single letter selection from a static two-dimensional letter matrix. All seven GOMS statements are executed in this path, which requires one cognitive cycle (τ_c) for each statement, with additional time for one mental *Operator* (t_1), representing statement (1.3) and the letter selection *Operator* (t_2), representing statement (2.2). For a word of length L, therefore, the estimated performance time for this interface is: $T = (L)(7\tau_c + t_1 + t_2)$.

The time for the mental *Operator,* t_1, represents the time to decide whether or not to continue with text generation. It is not easily subdivided into its component steps. The time for the letter selection *Operator,* t_2, represents the additional time required to do a single *row-column selection:* $t_2 = (d)(s) + 2h$, where: s = system scan rate (i.e., speed at which selection options are presented); d = average number of scan steps required to select an individual letter; h = switch hit time (i.e., how fast the user can hit the switch), with two switch hits required in row-column scanning. The variable d can be determined by knowing the position of each letter in the matrix and the probabilities of selecting each letter. The value of d is 2.245 with the frequency-ordered letter matrix used. This equation does not include visual search time, because if it is assumed that the letter matrix is well-learned and the scan rate is set appropriately, visual search can be done during the scan delay, so it does not require any additional time (Horstmann, 1987).

The PACA_2 System. The GOMS model for the PACA_2 system contains 29 statements. There are four possible paths through the PACA_2 system model:

1. Single letter selection without a search of word prediction lists.
2. Single letter selection following an unsuccessful search of both prediction lists.
3. Word selection when word is found in first prediction list.
4. Word selection when word is found in second prediction list.

For every word, the first two letters must be selected, so path No. 1 is executed twice per word. The time required to complete the word depends on the prediction algorithm, which determines the relative frequencies of each of the remaining paths. Therefore, the average time, T, required to generate a word of length L can be expressed as:

$$T = (2)(T_1) + (1 - p)[T_2 + (L\text{-}3)(T_1)] + (w_1)(p)(T_3) + (w_2)(p)(T_4)$$

where: T_i = time required to execute the ith path; p = probability that word is in the dictionary; w_1 = probability that word is on 1st prediction list, given that it is in the dictionary; w_2 = probability that word is on 2nd prediction list, given that it is in the dictionary.

Using the GOMS model, expressions for the T_i terms can be formed. For path No. 1 (letter selection without word search), 14 GOMS statements are executed, plus time for three mental *Operators* and one letter selection *Operator.* Therefore, the estimated time for path No. 1 is: $T_1 = (14)(\tau_c) + t_1 + t_2 + t_3 + t_4$.

Similarly, the estimated execution times for the remaining paths are:

$$T_2 = (40)(\tau_c) + t_1 + t_2 + (2)(t_5) + (2)(t_6) + (2)(t_7) + (2)(t_8) + t_3 + t_4$$

$$T_3 = (19)(\tau_c) + t_1 + t_5 + t_7 + t_2 + t_9 + t_4$$

$$T_4 = (31)(\tau_c) + t_1 + (2)(t_5) + (2)(t_7) + t_8 + t_6 + t_2 + t_9 + t_4$$

where: t_1 = time to decide if current selection is the first or second letter in the word; t_2 = time to determine if a letter or a word is going to be selected; t_3 = time to select from letter menu; t_4 = time to decide if all desired text has been entered; t_5 = time to search word menu; t_6 = time to select 'Next" from word menu; t_7, = time to decide if target word was found; t_8 = time to decide if more word lists remain to be searched; t_9 = time to select from word menu.

The PAL System. With the PAL system, if the user searches the prediction list after every letter selection, there are only two possible selection paths, as follows:

1. Letter selection after deciding that the desired word is not in the prediction list. (T_1)
2. Word selection when the desired word is found in the prediction list. (T_2)

However, if the word is not present in the prediction list after the third letter selection, it is almost certainly not going to appear in subsequent lists, particularly with a dictionary of only 1000 words (Jones & Wepman, 1966). Therefore, it is assumed that the user does not search the prediction lists that appear following the fourth and subsequent letter selections. When selecting the fifth through last letters in the word, the user simply selects letters without searching the prediction list. This is shown below as T_3.

The number of times that each of these paths are executed per word depends completely on the characteristics of the prediction algorithm. Representing the prediction parameters is much more complex than the PACA_2 system because the prediction list is

searched before each selection, rather than only after the second letter as in the PACA_2 system. If x_1 is the probability of successful word prediction following selection of the ith letter, the average time required to select a word of length L is:

$$T = [(x_0)(T_2) + (1 - x_0)(T_1)] + (1 - x_0)[(x_1)(T_2) + (1 - x_1)(T_1)]$$
$$+ (1 - x_0)(1 - x_1)[(x_2)(T_2) + (1 - x_2)(T_1)]$$
$$+ (1 - x_0)(1 - x_1)(1 - x_2)[(x_3)(T_2) + (1 - x_3)(T_1)]$$
$$+ (1 - x_0)(1 - x_1)(1 - x_2)(1 - x_3)(L - 4)(T_3)$$

Using the GOMS model, expressions for the T_i terms can be formed. These can be expressed as follows:

$$T_1 = (24)(\tau_c) + t_1 + t_2 + t_3 + t_4 + h + t_5 + t_6$$

$$T_2 = (24)\tau_c) + t_1 + t_2 + t_3 + t_4 + h + t_7 + t_6$$

$$T_3 = (16)(\tau_c) + t_1 + t_4 + h + t_5 + t_6$$

where: h = time to hit the switch for scan initiation; t_1 = time to decide if current selection is at least fourth letter of word; t_2 = time to search word menu; t_3 = time to decide if word was found; t_4 = time to decide if selection is a letter or word; t_5 = time to select from letter menu; t_6 = time to decide if all desired text has been entered; t_7 = time to select from word menu.

APPENDIX B

MODEL INPUT PARAMETERS FOR MODEL SIMULATION

The process of choosing values for the model input parameters is detailed below. These parameters are:

Basic Processor Times

- cognitive cycle time
- perceptual cycle time
- motor cycle time

System Parameters

- system scan rate
- average number of letters/word
- average number of scans/word selection
- prediction success parameters

Operators

- switch hit
- word found
- selection is letter or word
- 1st or 2nd letter of word
- at least 4th letter of word
- search list for word
- decide if text is complete

Values for the Cognitive, Perceptual, and Motor Processor cycle times (τ_c, τ_p, and τ_m) are taken from basic human information processing research (Card, Moran, & Newell, 1983). All three values can be estimated at 0.1 seconds for people without cognitive, perceptual, or motor impairments. These values were used for initial simulation trials as they also represent a wide range of users with disabilities whose cognitive, perceptual, and motor times are identical to able-bodied individuals, when using a one or two-switch scanning interface.

All except one of the mental *Operator* times are estimated by determining the relative number of component Processor times. For example, the time required to determine if the current selection is the first or second letter of the word can be decomposed into the following Cognitive Processor steps. First, place pending selection in short term memory. Then match it against first letter of word, then against the second letter of the word. These three steps take three Cognitive Processor cycles, or 0.3 seconds.

The text-complete *Operator* is one that cannot be readily subdivided into component Processor cycles. Therefore, a value of 1.35 seconds was used, taken from Card, Moran, and Newell's study (1983) of the generic "mental" Operator, M. This estimate is used for lack of a more refined estimate.

Time required to hit the switch can be modeled as a simple reaction time once the desired highlighted letter has been perceived. This reaction time takes one cognitive cycle and one motor cycle, or 0.2 second. The minimum scan rate can be set at the time it takes to perceive a letter on the display and match it to an image of the desired letter, plus the switch hit time. This is the sum of τ_p, τ_c, and 0.2 or 0.4 seconds.

There have been many studies to determine the average number of letters per word in samples of the English language. Two values that are frequently cited are 5 letters/word (Goodenough-Trepagnier et al., 1982) and 5.7 letters/word (Kucera & Francis, 1967). Five letters/word was chosen as the nominal estimate for simulation trials.

An accurate estimate of the number of scan steps per word selection cannot be made without explicit knowledge of the content of each prediction list and the frequency distribution of selections from each prediction list. For a frequency arranged list, the true number would be something less than one-half the number of words in the prediction list. A high estimate of one-half the number of words in the prediction list is used as the nominal parameter.

Similarly, precise values for the all prediction success parameters cannot be obtained without exact knowledge of the dictionary contents. For the PACA_2 system, the developers report a 0.70 probability of a word being present in the dictionary, but they don't specify the frequency distribution between first word list as compared to the second (Gibler & Childress, 1982). Since the word lists are frequency-arranged, it can be assumed that the probability of a dictionary word appearing on the first list, w_1, is greater than the probability of a word appearing on the second list, w_2. Therefore, nominal estimates of 0.70 and 0.30 were used for w_1 and w_2, respectively.

The estimation process for the PAL system is more complicated. Again, the developers provide a value of 0.75 for the probability, p, of a word being present in the dictionary, but the probabilities of a word appearing in the list after the ith letter selection are not given (Arnott et al., 1984). Therefore, two assumptions based on word frequency studies were used to estimate these probabilities. First, the probability of a new word appearing in the word list after the fourth letter selection was assumed to be zero. This means that the only required estimates are for x_0 through x_3. In addition, the relationship between the X_i's and p means x_3 is uniquely specified, given p, x_0, x_1, and x_2. The second assumption is that the words in the list before any letters have been selected are the 10 most frequently used words in the English language. According to Kucera and Francis (1967), these words are used 25% of the time, which yields an estimate of 0.25 for x_0.

The only remaining estimates needed are for x_1 and x_2, the prob abilities of a new word appearing in the prediction list after the first and second letter selections. One constraint is that $x_2 \geqq x_1 \geqq x_0$, since the predictions are refined after every letter selection. The only other constraint is that $x_3 > 0$. In a dictionary of 1000 words, adding a third letter to the word does not substantially improve the chance of a successful word prediction (Jones & Wepman, 1966). So, the nominal estimates used for the PAL system prediction parameters, x_0, x_1, x_2, and x_3 were 0.25, 0.35, 0.40, and 0.15, respectively.