Department of Computer and Information Science
University of Massachusetts
Amherst, Massachusetts 01003

The Speakeasy Communication Prosthesis System

Thomas Gruber

COINS Technical Report 84-22

August 1984

Abstract

Communication prosthesis is a means by which profoundly disabled, nonvocal individuals, for whom ordinary modes of communication are difficult or impossible, may express themselves. Conventional prosthetic techniques are inflexible and limited in coverage, and require special skills of the listener and speaker. Computer prosthesis promises to address these shortcomings, and to provide a powerful new technology for serving the communication needs of the disabled. The most important feature of a computer prosthesis system is the user interface. A model of a user interface for computer prosthesis is presented in which the computer takes the role of an *intelligent communication assistant*. The assistant presents information to the disabled user, who makes a choice by giving a unitary response (using a very simple and inexpensive device). The assistant interprets the response in the context of the interactive session, and then generates output which conveys the user's meaning. The details of the model are described in terms of the information flow which occurs between the user, assistant, and the outside world. The model is implemented in a computer program called SpeakEasy designed as a real world application in communication prosthesis.

Many of the ideas in this document have arisen from discussion with Stan Kulikowski II of the University of Massachusetts and Bruce Hawkins of Smith College, who developed the original communications prosthesis software.

1. INTRODUCTION

1.1 The Need for Communication Prosthesis

The ability to effectively communicate is essential to participation in modern society. Yet a large number of people lack the physical abilities to communicate without special assistance. People who are nearly paralyzed, such as children afflicted with cerebral palsy or adults who have suffered high spinal injuries, may be incapable of fluent speech, sign language, or writing. They require some kind of *communication prosthesis* to help them convey their meaning.

1.2 Conventional Prosthetic Techniques

In general, the goal of a communication prosthetic can be seen as a means to both improve the medium and the message. Since the severely disabled person is deprived of the normal media (speaking, signing, writing), a prosthetic technique must make the best possible use of a more limited means of expression. Improving the message requires extracting the maximum meaning from the limited medium. A variety of prosthetic techniques have been developed for this purpose. The following descriptions are incomplete and simplified, but they are intended to provide a context in which to consider computer prosthesis.

1. **Twenty Questions:** A skilled human assistant verbally prompts the disabled person, who can respond in some limited way to indicate a yes/no response. The assistant must be able to unambiguously "read" the response, and more importantly, provide a context in which the binary (yes/no) response can make sense. The assistant must anticipate, to some extent, what the disabled person wants to talk about, and must be able to choose prompting questions which can narrow the range of possible interpretations quickly. This method is clearly *knowledge intensive*. Its success depends on how much the assistant knows about the kinds of things that the disabled person would like to discuss. The world of possible things to talk about is called the *domain of discourse*. It is extremely difficult for the disabled person to express novel or unpredictable meanings, since they must lie outside the domain of discourse that the assistant is familiar with. Some method for composing meanings from known primitives, like "spelling it out", is used in cases where the intended message cannot be anticipated. Note that under these conditions, this method loses the benefit of the assistant's knowledge. The twenty questions method is the basis for the intelligent assistant model of communication prosthesis presented here, with the important difference that the assistant is played by a computer program.

2. **Language boards:** A language board consists of a set of items drawn on a physical surface, like wood or cardboard. The disabled user communicates by pointing, in some way, to the items on the board; the assistant watches and combines the items to form an utterance. Usually the items are words and simple phrases. If the user can spell, a matrix of characters is also provided. The pointing can be difficult for the user, and the assistant must be able to "read" the typically shaky response. The use of language boards is limited by the pointing task and the inherently finite vocabulary that a physical surface can support.[1] However, knowledge of the domain of discourse can be incorporated into the layout of the language board. The optimal spatial arrangement of items is nontrivial and many of the presentation techniques incorporated in the intelligent assistant model are derived from experience with the use of language boards.

3. **Special purpose hardware:** It is often possible to build special devices which help a disabled person overcome a communication handicap by providing an alternate means of language generation. A custom-built device might enable the user to control a typewriter-like output machine, which could generate written or spoken language. The contribution of the prosthetic is in improving the efficiency of the medium of communication. Special purpose hardware rarely considers the message: the vocabulary which can be generated is limited and difficult, if not impossible, to adapt to the individual user's needs. Even the parameters used to optimize throughput are sensitive to individual abilities; to configure a special machine to the needs of a disabled user is generally more difficult and expensive than to modify the database used by a computer program.

1.3 Computer-assisted Communication Prosthesis

The advent of inexpensive and powerful microcomputers has made many conventional prosthetic techniques potentially obsolete. The functionality of the conventional techniques is subsumed by the capabilities of machines which operate by following instructions encoded in programs and data. Most hardware operations can be emulated in software. For example, an early "computer" prosthetic, the Tufts Interactive Communicator (TIC), scanned rows and columns of characters by controlling a matrix of tiny lights. The TIC can easily be emulated with a very

[1] A knowledgeable assistant can often infer the intent of a message from a few items selected, so the user need not spell out the entire message. In this way, the user and assistant can make more efficient use of a finite vocabulary.

simple computer program, which can run on almost any microcomputer, even very inexpensive consumer models used primarily for entertainment. While it is difficult to modify the vocabulary of a language board or special purpose communicator (and it is impossible for the disabled user to do it), a computer program can easily update a custom database stored in the memory of the computer. The ability to adapt the vocabulary of a communication assistant is especially important for the disabled child, whose language development will require both new vocabulary terms and new ways of organizing them.

The microcomputer can be integrated nicely into the prosthetic environment of the severely disabled person. Disabled users use a myriad of devices to provide for personal needs. There may be a device for controlling a wheelchair, a simple switch for signaling personal distress, another interface for operating TV, stereo, and lights, and maybe another for directing telephone dialing equipment to call for help. All of these *control interfaces* require different responses of the disabled user. Some of these can be expensive (and so are often not available). Adding one more device, the microcomputer, to provide communication assistance may seem like more inconvenience. Ironically, the same microcomputer can provide the control interface for *all* of these environmental devices. The user need only learn one interface, which is general enough to subsume the others. Any improvement in the use of the computer prosthesis, through optimization performed by the program or just through practice, is automatically extended to all of the environmental devices. The net result is to simplify the life of the disabled individual.

There is an inherent limitation, however, to the use of a computer which is shared by all technological prostheses: *people are smarter than machines!* This fact is not a cause for pessimism; rather, it motivates much of the advanced research in computer prosthesis. Language is a complex process, requiring both *intelligence* and *knowledge* acquired over a lifetime. Most of us freely speak with each other without being conscious of the complexity of our means of communication. When a skilled assistant helps a nonvocal person communicate by asking questions and "reading" the responses, the assistant is bringing a vast amount of knowledge about the disabled person to bear: vocabulary, grammar, the current topic, expected answers, and even facts about the daily life of the person. No current man-made prosthesis can do that as well as humanly possible (sic), but research in artificial intelligence and computer science indicates that it is a goal worth pursuit.

2. A USER INTERFACE FOR COMMUNICATION PROSTHESIS

Our approach to communication prosthesis puts the computer in the role of *intelligent assistant*. The disabled user is capable of only a limited response; the computer must do the work of generating language from highly restricted input. In the proposed model, the user participates in an *interactive dialog* with a computer program. The computer presents a set of alternatives to the user. The set of alternatives is the computer program's "best guesses" at what the user wants to say or do. The user chooses from the alternatives by making a simple response, typically a binary response within an interval of time. The program then interprets that response, and generates output. The output is intended to be understood by other people, and is typically spoken or written language. This process can be described in terms of information flow through four interacting activities:

1. the **presentation** of information by the computer program to the user,
2. the **response** method by which the user indicates a choice from the alternatives presented, and
3. the **interpretation** of the user's choice, which may call for
4. the **generation** of output.

The functions of these four activities overlap, and they depend on each other for success. The presentation of alternatives will affect what the user can respond to, the response dictates the interpretation, and the interpretation directs the generation of output. For descriptive purposes, each activity will be considered in turn, and their interrelations will be noted as appropriate.

2.1 The Presentation of Alternatives to the User

The model of communication outlined here distinguishes assisted communication from normal speech by the inherent imbalance in the *bandwidth* of information flow between parties involved. Two able speakers are normally able to send the same amount of information to each other.[2] The communicationally disabled person, however, sends information on a channel with a reduced bandwidth. The computer

[2] People can comprehend spoken language faster than they can speak it, and one person usually speaks while the other listens, but they are both capable of interrupting and talking to each other at roughly the same rate. So both channels of communication between able speakers, one in each direction, operate at roughly the same bandwidth.

assistant must compensate by using its channel to present the user with the best possible choice of alternatives, so that the most can be made of each user response.

2.1.1 Enriching the Set of Alternatives

One set of alternatives to present to a literate user is an alphabet. Since any word can be composed of characters in an alphabet, total coverage of a language can be provided with a simple spelling set. While theoretically complete, an alphabet alone is obviously inadequate for the disabled person, who has difficulty choosing even a single character. The assistant must provide additional information to choose from. For instance, once the first character in a word is chosen, the computer assistant can provide a set of words which begin with that character as alternatives. If a second character is chosen, only words with the two character prefix need be offered, along with the spelling alphabet. This technique is an instance of standard hierarchical search methods used in computer science; if the user is "searching" for a word from a large vocabulary of words, the characters serve to reduce the "search space". Augmenting the set of alternatives (the alphabet) with subsets of the search space (words with matching prefixes) at each decision point (choice of next letter) is significantly more efficient than blind search (examining every possible word), or complete specification (spelling it out entirely). The simple case of spelling words demonstrates the principle that increasing the amount of information presented to the user can improve the overall rate of communication. Of course, the information to present has to be *appropriate*. If the set of words presented as alternatives was not a function of the user's previous choices, little improvement could be expected. This is especially relevant when the number of alternatives to present is quite large - too large to be presented as a set. Since this is usual for normal vocabularies, the choice of which subset of alternatives to present requires careful consideration.

2.1.2 Using Syntax to Constrain the Alternatives

Other methods for structuring the presentation can take advantage of the formal properties of the user's language. Natural language is rule bound, structured by syntax. The computer assistant can take advantage of this information to further constrain the search through the space of alternatives. If the user has chosen to create a declarative English sentence, (that choice is itself a large constraint), and has already chosen a noun, the assistant can present a set of verbs as likely next choices. Of course, there are many verbs to choose from, and only some are compatible with the given noun. The number and tense of subject and verb should match, so only those verbs with the proper conjugation should be presented. Unfortunately, grammar can only partially constrain the set of "next words" in a sentence.[3] One problem is that many people don't speak grammatically perfect English, especially in spontaneous conversation. Another is the complexity of grammar in natural language; nouns can be followed by almost anything in English! But more importantly, syntax alone misses the ultimate constraint in language - the meaning of the sentence.

2.1.3 Using Semantic Knowledge

An intelligent communication assistant can benefit from an understanding of the meaning of the objects with which the user may build an utterance. For example, if the user wants to talk about eating, the assistant can present a selection of words and phrases known to be relevant to food and consumption. Furthermore, the orchestration of the presentation of subsets of relevant items can be guided by the natural organization of the domain of interest; e.g., the assistant first presents places to eat and/or categories of food, then menus of particular kinds of food, and so on. The more information about the user's world that can be used by the assistant, the better the resulting communication. This is the source of communicative power for the human assistant: comprehensive knowledge about the disabled person's needs, desires, interests, habits, etc. can be used to help determine what the disabled person would want to say. It is difficult to capture the subtleties of human understanding in a computer, but research in knowledge representation has provided techniques for efficiently storing in a computer the kinds of knowledge that an intelligent communication assistant might use. Categorical and relational information are two sources of semantic knowledge which are easily accessible to the computer assistant.

2.2 Reading the Response from the Disabled user

The people for whom a communication assistant can do the most are both nonvocal and physically disabled to the degree that they are unable to control conventional input devices, such as a keyboard.[4] Disabilities vary in degree and kind, but the common characteristic of users of a communication assistant is that motor response is difficult, slow, and unreliable. Techniques for reading a response from a disabled user

[3] There is no reason why the individual items presented as alternatives to the user need to be single characters or words. It can be quite powerful to have an entire stream of thought - a joke, for instance - represented as a small number of "chunks", easily chosen by the user as units (the punchline could be a separate chunk).

[4] This is not to say that a person who can speak but lacks the use of his or her hands *cannot* benefit from an intelligent assistant; certainly for the purposes of written communication such a system would be invaluable.

must maximize speed and reliability and minimize fatigue and errors. Many input devices have been developed for this purpose, from sophisticated eye tracking systems to simple pushbutton switches closed by a movement of the head. They differ primarily on *bandwidth:* how many bits of information can be transmitted in a given unit of time. If reading a user's response is considered in terms of choosing from a set of alternatives, the techniques fall into two broad classes: *single input scanning,* in which the user can only make a simple binary (yes or no) response, [5] and *direction selection,* in which the user is capable of giving input on more than one dimension simultaneously, such as directing a wheelchair with a joystick (moving in two dimensions).

2.2.1 Single Input Scanning

This method only assumes that the user can reliably operate a single muscle. Often, a disabled person can operate much more than a single muscle, but due to neuromuscular dysfunction cannot reliably coordinate a group of opposing muscles required for a multidimensional response. Whatever the reason, all that is required for this input technique is that the user give a single response within some interval of time after presentation of a stimulus.

To translate a simple response into a *choice,* a communication assistant presents a set of alternatives and *scans* each one, until the user makes a response. A human assistant might point to each word on a language board, until the user makes a distinguishing movement, such as an eye roll or a nod. The word being pointed to at that time is considered the choice. Similarly, a computer assistant might move a "cursor" (a graphic pointer) over items displayed on a video screen and continually check the status of a switch controlled by the user. When the user closes the switch, the computer assistant considers the item under the cursor to be the user's choice.

There are several parameters of the scanning process which can be "tuned" to improve performance. Most obvious is the rate at which items are scanned, which is measured by the amount of time that the cursor stays on each item before going on to the next. Every user has differing abilities, so it is important that the scan rate be adjusted individually for each user. If it is too fast, the user will be unable to accurately select items. If it is too slow, the overall performance of the communication system will be degraded. The scan rate at which a user is comfortable may depend on more than just the actual speed of the cursor. When a

[5] It is more accurate to call the simple response unary, since what is sensed is either "on" or "no signal". When the unary response is controlled to occur (or not occur) within some time window, however, the effect is a binary decision at each choice point.

new set of alternatives is presented to the user, there is an interval of time required to read them. This is the *recognition period*. If the item which the user wants to choose is the first to be scanned, and scanning begins before the user can read all of the items, then that item will be missed on the first scan. If a proper recognition period is allowed for, the user can be prepared to respond to that first item when it is scanned. The familiarity of the set of alternatives can affect the length of the recognition period. If the user is very familiar with the current set, very little time is needed for recognition; if a set of alternatives is presented for the first time, more recognition time should be allowed. Variables such as the order of items (e.g., alphabetized or sorted by frequency), how the items are positioned on the screen (rows and columns, triangular matrix, high or low on the screen, space between items), number and complexity of items presented at once (more words than phrases?), and the presentation medium (icon or text, size of font or image, etc.) all presumably affect the optimal scanning rate for an individual user. The study of these variables and of individual difference in scanning performance is an area of research yet to be explored.

Another parameter of simple input scanning is what to do if the user fails to respond to any of the alternatives presented. There are two interpretations: 1) the user really wanted to select one of the items presented, but "missed" it when the cursor was scanning it, or 2) none of the items presented satisfied the user. There is no way that an assistant can know which interpretation to take unless the user explicitly supplies more information (which is not possible if the simple response paradigm is being used). An intelligent assistant must be prepared for both interpretations. One style of interaction allows the user N tries on the set of alternatives. The items on the screen are scanned repeatedly until one is selected or N iterations over the items have been completed. The lack of response is interpreted as the *null response*, which means "none of the above". Another style provides an explicit item to represent the null response, and can perhaps use the number of iterations over the items on the screen as feedback to the scan rate optimization process. For instance, if the user is letting the cursor pass by a desired item several times before selecting it, it may be a sign that the scan rate or recognition period parameters need to be adjusted.

The order in which the items are scanned can be important. If the user is to choose from a large number of alternatives, then those most likely to be chosen should be scanned first. If the items can be broken down into smaller groups, such as rows of items in a table of alternatives, then the number of scans required can be reduced; first, the rows are scanned (as alternatives themselves), and then the individual items in the chosen row are scanned.

2.2.2 Direct Selection

If the user can provide more than a binary response, then a communication assistant can read a response more efficiently. Consider an alphabet presented as rows and columns of characters. Simple input scanning, which requires only a binary response from the user, requires that the user first choose a row, and then a character on that row. This costs at least two responses and the scanning time needed to reach each selection. If the user could send both row and column information simultaneously, then the desired character could be reached directly. Examples of direct selection input include pointing (e.g., with an appendage or pointing stick), eye tracking, and servo control.

A familiar example of servo control is the "joystick" that is used to control many motorized wheelchairs. The joystick device allows for control in two axis, with movement off of center signifying an implicit velocity. Many disabled people have struggled to learn to control their wheelchair. These skills can be applied to direct selection, and the assistant can take advantage of the extra degree of control. For the spelling screen with an alphabet, the assistant could start with a cursor in the center of the array of characters, and instead of scanning the items, track the cursor up, down, left, or right in accordance with input from the joystick. The user can make a selection by either leaving the cursor on a particular item for some amount of time, or give some other kind of response, such as closing a switch, to indicate selection.

The performance of direct selection input methods can be affected by adjustable parameters, as in scanning. The tracking ratio of the cursor, the relation between the signals coming from the input device and the movement of the cursor, can be adjusted to the needs of individuals. The timing of item selection can also be tuned. The order of presentation of items used in a direct selection scheme can be optimized for performance. The items most likely to be chosen should be presented so that they are "closest" in response effort to the starting position of the cursor. If the user can control a cursor in two dimensions, then an optimal positioning of items would put the most likely items in the center of the screen, and surround them with less likely items in concentric circles.

There is a need for controlled experimentation to study the effect of control parameters on user performance. An intelligent assistant can do automatic optimization of multiple parameters if it is given some model of how they might affect performance. The relative merits of simple scanning and sophisticated direct selection schemes is still not known to sufficient detail. Fortunately, most profoundly disabled users can usually give enough of a response to use scanning, and direct selection users can certainly use a scanning system. For this reason the scanning method, when tuned to individual needs, is a good candidate for a *universal prosthetic input method*.

2.3 Interpreting User Input

After a communication assistant has obtained input from the user, it has to interpret the response. Recall that the response is the selection of an item from a set of alternatives presented to the user. The normal interpretation of an item which is already in the form of text is to simply output the item as is. If the user selects the word "scratch", then the assistant would simply send "scratch" to an output device (see the next section for a discussion of the use of output devices). Such an item is called *terminal* because interpretation of the item ends with the lexical value of the item itself; in this case, the interpretation is nothing more than the word "scratch". This is the behavior of a simple communication prosthetic, but an intelligent assistant can perform a variety of actions corresponding to user requests. The sophistication with which the user can control the communication process is determined by the range of actions which the assistant provides.

2.3.1 Nonterminal Items: Representing Database Relationships

One type of action is the interpretation of *nonterminal* items. A single nonterminal item is associated with a set of other items; it "stands for" the whole group. The action performed when a nonterminal item is selected is to present the group represented by the nonterminal as the next set of alternatives to choose from. The item presented as "FOOD", if selected, might cause a whole screen full of items about food to appear. In most cases, the nonterminal item itself does not become part of the output stream. Nonterminal items can be used to categorize words and phrases by their meaning or syntactic role. For example, a "menu" of nonterminal items with the following items could be presented: NAMES, MEALS, CLOTHES, ASSISTANCE. When the user chooses MEALS, for instance, the word MEALS is not sent to the output stream, but rather another set of items about meals appears: BREAKFAST, LUNCH, DINNER, SNACK. This is in turn a menu, which leads to further categorization. The nonterminal item allows for an arbitrary network of sets of alternatives, which enables the user to direct the assistant to present specialized sets of items. This is a very powerful capability, as it can be used to organize a large amount of customized information germane to the communication task of a particular user.

The selection of a nonterminal item is one way that the user can give *control information* to the assistant. Control information tells the assistant what to do next, i.e., what set of items to present. A database of items can be related in arbitrarily complex ways, containing hierarchical relationships (the food menus are an example), associations links, syntactic relations and roles, etc. An assistant can provide actions to allow the user to travel through the database along all of these dimensions. Grammars can be built up by providing *paths* among sets of items which follow the rules of syntax. The simple declarative sentence might follow the syntactic rule

SUBJECT VERB OBJECT. In this case a set of subjects is presented first, followed by verbs, and then objects. Semantic relationships among subject, verb, and object can be determined by paths between specific items. If the set of verbs following the choice of "I" as the subject is {AM, WANT, GIVE UP, LOVE}, then each could get a meaningful set of objects to follow it. Following AM might be words like HUNGRY, HAPPY, SLEEPY, after WANT might be A SNACK, TO LEAVE, SOME HELP, whereas GIVE UP would have no object items following. From these examples it can be seen that there is a lot of information contained in the relationships among items. The assistant can store these kinds of relations in addition to its basic vocabulary in the database for a particular user.

2.3.2 Control Actions: Telling the Assistant What to Do

The action associated with an item may perform a task which is not directly related to the generation of language. Actions in this class are called *control actions*.[6] One type of control action allows the user to modify the parameters of the input interface. A special item for setting the scan rate, for example, could cause a set of speeds to be presented. When the user chooses one of the speed items, the scan rate is immediately altered. Another type of control action provides for error correction. An "oops!" item, for example, does not generate language, but instead recovers from the most recently selected action. A backspace or rubout key on a word processor or error correcting typewriter performs this function by erasing the last character typed.[7] Other control actions could give the user control over the output devices. An example is an item which causes the current message to be sent to the printer. There are many more special tasks that can be performed by control actions; in fact, the communication assistant can be a disabled user's interface to an entire computer system. This implies that *anything that an able user can do with a computer can be made accessible to the disabled user*. The possibilities are exciting: word processing, musical composition, business forecasting, electronic mail, and, of course, computer programming.

[6] The term "control" is slightly misused here, because there is control information encoded in nonterminal item actions. Control actions provide means to control the behavior of the assistant independent of the particular lexical values of items.

[7] In general, error recovery can do more than just erase the last piece of generated text. Each action for which the assistant has an interpretation can be associated with an inverse action, which nullifies the effect of the action if it were erroneously selected. The inverse of an action which changes the scanning rate to a new value would return the scanning rate to the previous value, for example.

2.3.3 A User Modifiable Knowledge Base

A major limitation of current communication prosthesis systems is the difficulty to grow with the user. The importance of a customized knowledge base for each user is clear, but the question remains: how is that knowledge acquired? An intelligent assistant should be able to accept new knowledge as it operates. The person best qualified to teach the assistant is the person most familiar with the user's communicative needs, the disabled user. A powerful capability of an intelligent assistant is the ability to incrementally accept new knowledge in its domain. So as part of the interpretation of ongoing user responses, the assistant should be able to incorporate knowledge about what the user is likely to say in the future. Techniques for achieving this can range from *declarative database updates,* in which the user explicitly builds knowledge structures used by the assistant, to *computer learning,* in which the assistant takes an active role in inferring its own knowledge structures based on the response patterns of the user. This area of research is still in its infancy, and progress can mean significant improvements in the communication prosthesis technology.

2.4 Generating Communicative Output

The final task of a communication assistant is to send messages to the outside world which convey the meaning intended by the disabled user. If another human being is physically present, then the disabled person and communication assistant work as a team, with the assistant acting as mouthpiece. That mouthpiece can be implemented on any of several output devices which can be controlled by a computer. Most common is the display screen, conveniently the same screen used by the computer to present alternatives to the user. This has the advantage of being fast, quiet, and inexpensive, but the disadvantage of being an unnatural medium of communication between two human beings.

Speech synthesizers can also be used. They can generally take an arbitrary stream of text, like the display screen, and convert it into speech; the quality of the speech depends on the particular synthesizer used. One way to make synthesizers more understandable is to send them phonetic sequences instead of raw text; the phonetic "spellings" of lexical items generated by the communication assistant can be precomputed and stored. A similar technology can take prerecorded segments of digitized speech and generate high quality output. The vocabulary of a nonvocal person can be spoken by a vocal person and stored, and the communication assistant can retrieve the recorded speech and play it back as directed by the user. This produces the most natural form of communication for the disabled person. A five year old disabled girl can be given the "voice" of another five year old girl whose speech has been digitized and incorporated into the disabled user's communication database.[8]

Other output devices provide more permanent storage of the user's message. A printer can generate paper copy, which can be read by people physically separate from the user (e.g., teacher, employer, friend). Or the output of a communication assistant can be stored on standard computer storage devices, such as magnetic disks, and easily retrieved later by the user. Since the generation of text is difficult and slow for a disabled user, even with the aid of an intelligent assistant, the ability to efficiently store and retrieve text can let the disabled user work alone to prepare a lengthy message and call it forth in "real time" when an audience is present.

One output technique combines advances in computer and telecommunications technologies with that of communication prosthesis: *telephone networking*. It is possible to connect computers directly to telephone networks, and to send voice or machine-readable data over the lines to other computers or people. With the use of voice synthesizers, a communication assistant can send messages from nonvocal user to a person listening on the phone at the other end. Or if the other person also has a computer connected to the phone line, generated text can be transferred and then displayed, printed, spoken, or stored at the remote end of the phone line. The intelligent communication assistant technology opens the possibility of a group of communicationally disabled people actually talking over the telephone!

[8] The use of artificial speech can open up fascinating new areas of research for those interested in treating the speech pathology of severely disabled people. Kulikowski has speculated on the use of a synthesizer with disabled infants as early as the synthetic babbling stage. It might be possible for an infant to associate synthesizer sounds with symbols on a control interface and use them to form syllables as a normal baby would do with its own babblings.

3. THE SPEAKEASY PROGRAM

SpeakEasy is a computer program which implements the model of human interface for communication prosthesis described in previous sections of this document. Its purpose is to compensate for communication disabilities, by doing the bulk of the low-level work of *generating output*, while giving the user maximum control over the high-level task of *deciding what to output*. The disabled user communicates with SpeakEasy through a simple input medium, usually by closing a simple switch. SpeakEasy interprets the response as a selection of one of a set of items which it has presented to the user on a video screen. It then generates speech (via a speech synthesizer), text displayed on a video screen, or machine-readable data which can be stored in files on permanent storage and printed on paper. The process of item selection from limited input and the control of output devices are trivial tasks for a computer program. The major contribution of this work is the implementation of the interface between the human user and the computer program which enables it to efficiently map the limited responses of the severely disabled user to natural language text conveying a meaning to another person.

SpeakEasy is written in a portable high level programming language and is designed to run on most commercially available microcomputers It is designed to be easily adapted to new hardware and software environments. Like most large computer programs, SpeakEasy is decomposed into several semi-independent modules, roughly corresponding to the functional components of the model of user interface previously described: presentation, response, interpretation, and generation. The description to follow assumes a familiarity with the model, and concentrates on the details of implementation more than theoretical issues.

3.1 The Knowledge Base

Underlying all of the processes in the user interface is a database which contains the information that SpeakEasy uses to interact with the user. All of the linguistic and control information available to the SpeakEasy program is represented explicitly in a hierarchically structured knowledge base. The program is said to be "data driven," meaning that its behavior is largely determined by the data in its knowledge base. For example, the program itself knows nothing about language and certainly nothing about what individual users want to say; that information is provided for each user in the knowledge base. The set of actions which the user chooses to use, their names, and the context in which they are made available is also specified in the knowledge base. The most general level of the hierarchy contains *global parameters*. Global parameters include the optimal settings for interaction parameters (like the optimal scan rate for the user), the hardware configurations of the computer (input/output devices, memory restrictions, etc.), and an interface to the file system of the host operating system. The next layer of the knowledge base is divided into *subnets*, which are independent networks of linguistic and control items. Each subnet contains a logically coherent set of knowledge about a particular domain; for example, there may be a subnet which has information about eating and food. Each subnet is made up of a set of *lists*, which are groups of items presented to the user. Each *item* contains a lexical representation, that is, a string of characters presented to the user, formatting information specifying how that item is to be output (i.e., should it start on a new line, should it be separated from previous items by a blank), and an action to be performed if that item is selected. There is default formatting and action information at both the list and subnet levels.

The database is currently stored as a set of ASCII text files, compatible with most operating systems and networking protocols. A knowledge base can be created and modified with the use of any text editor.[9] The format of the files allows for extensive documentation; in fact, one set of example database files serves as a "learning by example" tutorial to database construction. Of course, SpeakEasy can be used to create database files directly, so disabled users of the program can potentially modify their own databases. At present this is a task requiring specialized training, but it is anticipated that a user's knowledge base can be automatically adapted to patterns of use. There are several ways that the disabled user can modify databases directly with SpeakEasy (as opposed to the program using computer learning algorithms to deduce the user's intentions). The user could explicitly instruct SpeakEasy to store some previously constructed fragment of text. The program could automatically analyze generated text and extract words and phrases to present as complete items in the future. A database construction subnet could give the user an interface for editing the personal database. All of these methods need to be investigated if they are to be evaluated for efficiency and ease of use.

3.2 The Display Presentation System

Lists of items are always being presented to the user. Sometimes there are more items in a list than will fit on the display screen. Often it is desirable to present more than one list of items on the screen at a time, to provide a sense of continuity over selections. To provide for these needs the display system performs dynamic item formatting. The screen is broken up into "windows" - rectangular areas which can be written to as independent virtual output devices. One such

[9] This mode of database editing assumes an ability to use a text editor (word processor). Normally, this would be done by an able-bodied assistant, teacher, family member, or friend of the disabled individual. However, since SpeakEasy can provide the functionality of a text editor, disabled users can also edit knowledge bases this way, although with less efficiency.

window is the text area, which presents the previously generated text as a text editor would. The selection area is a set of windows which contain the lists being presented for scanning. In each window of the selection area a list is dynamically formatted; the number and length of the items in the list are independent of their presentation on the screen. The set of lists on the screen at any given time depends on the contents of the lists themselves, but typically they form a sequence of default choices. Consider the screen display of figure 3.1:

```
Choose a NOUN:

I                   WANT                TO HAVE LUNCH
YOU                 LIKE                TO GO OUTSIDE
THEY                HATE                TO EAT
JOHN                AM                  <PEOPLE>
MARY                WILL                <EMOTIONS>
MOM
```

Figure 3.1

The first column is a list composed of items which are all nouns, suitable as subjects of a simple sentence. The second column is a list of verbs, and the third a list of objects. The system is shown scanning the first column. Although the user must choose an item from the first column before any from the other columns, the others are displayed to provide context for the selection. When an item from the first column is chosen, the next lists to scan are often changed by the action associated with the chosen item. For instance, if YOU is chosen as the subject, the next list would be replaced with one which has the correct verb agreement, i.e., with ARE instead of AM. In this example the entire sentence "I want to have lunch" is displayed at once on the screen; it is the default choice for the three lists.

A third area of the screen contains a special list of items which is always presented in addition to the lists in the selection area. This is the *control list*, and it contains special control actions which are always applicable regardless of the current items appearing in the selection area. Examples include UNDO, SPELL, SPEED SET, and RETURN.

3.3 The Input System

The user's response is defined as the selection of an item from a list of presented items. For the severely disabled user, a scanning technique is used, moving a cursor over items in a regular pattern until a a response is sensed. The response is measured as the closure of a switch which the user can reliably control. The item which is currently pointed to by the cursor when the switch is closed is interpreted as the user's selection. The rate at which the items are scanned is controlled by a parameter which can be changed by the user. A direct selection input technique is also provided for users who are able to operate a keyboard. SpeakEasy is designed to accommodate new input techniques as they are developed.

3.4 The Action Interpreter

The action interpreter is the component of SpeakEasy which performs the operations which the user selects by choosing items in the knowledge base which are associated with actions. This function roughly corresponds to the part of the user interface model called "interpreting user input," except that the generation of text to be sent to output devices is handled by the output stream management system to be described shortly. SpeakEasy provides a set of primitive actions which are implemented directly by the program as subroutines, but are accessible to the user of the system via items. All items can be associated with a named action; when an item is selected, its action is performed (e.g., a subroutine is executed). The set of actions available determines the possible behaviors of the system.

The initial set of actions include operations to traverse the hierarchy and relations of the knowledge base. They allow the selection of an item to tell the system which set of items to present for the next selection. In the example of the simple sentence presented in figure 3.1, the item YOU would be associated with an action which would change the next list of items to present, in this case, a list of verbs which agree with the second person subject.[10] In addition to simple actions, special control actions give the user control over the parameters of the input and output algorithms, and implement the primitive operations of "application" subnets, such as an editor or test administrator. The program is designed to be easily extended; special actions are trivially associated with new subroutines which can perform arbitrary computation. Some primitive actions include:

[10] The control operations are powerful enough to simulate context free grammars and many context sensitive grammars. They are like a primitive programming language in that facilities for sequential flow of control, conditional branching, and recursion are supplied.

- The null action: do nothing.
- Go to a new list: present a new list to be scanned.
- Call a new list: present a new list, but remember the current list.
- Return to a previous list: return to a list remembered by a call-list action.
- Go to a new subnet: start scanning an entirely new subnet of lists.
- Call a subnet: analogous to call-list.
- Set the scan speed: change the current rate of scanning items.
- Undo the previous action: recover from a mistake by reversing the effects of the previous item selected. This is a very powerful action, since it allows for robust error handling capabilities.

3.5 The Output Stream Management System

The output of SpeakEasy is text. Although the text can be sent to a variety of output devices, such as printers and speech synthesizers, the underlying datum is a stream of characters. Managing strings of characters in a computer is a well understood process, and the canonical application is text editing (also known as word processing). SpeakEasy supports the capabilities of a modest screen-oriented text editor: insert and delete, move a cursor and a window over a file, save and retrieve files to and from disk storage, and primitive formatting (wrapping lines of text which do not fit on a single line of the screen without destroying the continuity of words). SpeakEasy can thus be used by students and professionals to create and edit large documents, despite a disability which would prevent them from any written communication without assistance. The disabled user has full control over the disposition of text that she has carefully constructed. If in a face to face conversation, the user can instruct SpeakEasy to send messages to a speech synthesizer, to be spoken out loud. If she is alone and preparing a document to be presented in writing, the user can have the program print out a copy on paper. If the computer is connected to a telephone, the text can be sent to another SpeakEasy user on the other end, or to a timesharing computer (perhaps at work or school). Like the input system, the output stream management system is designed to be easily extended to new applications.

3.6 Summary

SpeakEasy implements the model of user interface for communication prosthesis described in previous sections of this document. The presentation system takes maximum advantage of the computer screen to offer a set of items in context for the user to see. The input system implements parameterized scanning and direct selection algorithms to provide flexible response methods accessible to the severely disabled. The action interpreter supports a powerful and extensible command language for applying the structured knowledge base to the text generation task. Text can be sent to a variety of output and storage devices, all under the direct control of the disabled user. SpeakEasy is an intelligent assistant, providing the means for a severely disabled nonvocal person to communicate with the rest of the world. Now, if they will only listen...

References

BYTE, 7(9), September 1982. Special issue on computers and the disabled.

Goldenberg, E. Paul. *Special technology for special children*, Baltimore: University Park Press, 1979.

Goodenough-Trepagnier, C. and Rosen, M. J. An analytical framework for optimizing design and selection of nonvocal communication techniques. *Proceedings of the International Federation of Automatic Control Conference on Control Aspects of Prosthetics and Orthotics*, Columbus, Ohio, May 6-9, 1982.

IEEE Computer Society, *Computer*, 14(1), January 1981. Special issue on computers and the handicapped.

Kernighan, Brian and Ritchie, Dennis. *The C programming language*, Englewood Cliffs, NJ: Prentice Hall, 1978.

Mann, W., Bates, M., Grosz, B., McDonald, D., McKeown, K., and Swartout, W. Text generation: the state of the art and the literature. USC/Information Sciences Institute report ISI/RR-81-101, December 1981.

Robertson, G., McCracken, D., and Newell, A. The ZOG approach to man-machine communication. Department of Computer Science, Carnegie-Mellon University, CMU-CS-79-148, October 1979.

Vanderheiden, Gregg C. *International software registry of programs written or adapted for handicapped individuals.* Trace Research and Development Center, University of Wisconsin, 314 Waisman Center, Madison, Wisconsin, 53706.

Woods, W. A. Transition Network Grammars for Natural Language Analysis. *Communications of the ACM*, 13(10), October 1970, 591-606.