A User Interface for Deaf-Blind People (Preliminary Report)

Richard Ladner, Randy Day, Dennis Gentry, Karin Meyer, Scott Rose

Department of Computer Science
University of Washington
Seattle, Washington 98195
ladner@cs.washington.edu

Abstract

A user interface suitable for deaf-blind users is presented and justified. The interface is designed for small paperless Braille displays, large font visual displays, or other low-bandwidth displays. Some of the key properties of the interface are that it uses a hierarchical approach to structure both commands and data, has a small universal command set, and has pervasive editing capability. DBNet, a system employing the user interface, has been built and tested with deaf-blind users. DBNet will provide various communication services to the deaf-blind community including electronic news, mail, and bulletin boards.

Résumé

Un interface homme-machine adapté aux usagers sourds et aveugles est motivé et presenté. L'interface est adapté pour de petits claviers de Braille, des écrans d'affichage avec de larges fontes de caractères ou d'autres modes d'affichage utilisant une communication à faible largeur de bande. Quelques propriétés essentielles de cet interface sont une approche hiérarchique pour les commandes et données, un petit jeu de commandes universelles et une possibilité d'éditer en tout circonstance. DBNet, un système qui emploie cet interface, a été construit et essayé par des usagers sourds et aveugles. DBNet fournira divers services de communication à la communauté des sourds et aveugles, tels que du courier et des nouvelles électroniques ou un accès à des services d'information.

1 Introduction

The trend in computer user interfaces seems to be toward more graphics and voice and less plain text for both input and output. High quality bit-mapped displays and speech synthesis output devices have created an enormous potential for new user interfaces. Research centers and commercial products are already effectively using this potential. However, there is at least one group of users that is being left out of this revolution. This is the group of deaf-blind people, people who can neither hear nor see. Graphics and speech are useless for this group. This paper addresses the questions of what use computers are to deaf-blind people and, assuming they can be useful, what is the best way to provide computer access to this group of users.

Although the number of people who are simultaneously both deaf and blind is very small (estimated at about one in 10,000), we believe that this group can benefit tremendously from computer-assisted communication. Most of us have easy access to other people and information through everyday speech, telephone, newspapers, magazines, television, and radio. Unfortunately, these media are not easily accessible to deaf-blind people, with the consequence that deaf-blind individuals have tended to lead isolated lives with limited human contact, to be poorly educated, and to lack knowledge about current events. There is a slow, but noticeable, improvement due to the emergence of communities of deaf-blind people in some cities like Seattle, the opening up of educational and job opportunities, and the development of new devices such as the TeleBraille, a paperless Braille terminal for telephone communication.

Our goal is to improve the lives of deaf-blind people even more by increasing their access to information and other people. It is our belief that general improvements in communication using computer networks can make a significant positive impact [1][2][5]. Our plan is for each deaf-blind person to have a personal computer in their home equipped with a suitable output device: Braille, large font, Morse code, etc. These personal computers would be networked with each other and, through a gateway, be connected with other existing networks. Using their home computers, deaf-blind people could receive the daily news and weather electronically or correspond rapidly with their friends, employers, and service providers using electronic mail. Such a network would enrich the growing community spirit among deaf-blind people. While a home computer employed as an information center is not a new idea, the problems inherent in making this technology available to deaf-blind people are formidable.

A key problem in developing a computer network for deaf-blind people is designing a user interface that they can learn and use easily. In this paper we present, justify, and report on experience with a user interface that is suitable for use by deaf-blind individuals. We describe DBNet, an integrated family of communication applications, which employs the user interface we have designed.

This work is supported in part by a grant from IBM Corporation.

TeleBraille is manufactured by Telesensory Systems, Inc., Mountain View, California

2 User Interface Issues

A number of important issues influence the design of a user interface for deaf-blind people. First, the intended user of the interface has certain capabilities and limitations that must be kept in mind. Second, there are only a limited number of input/output devices which deaf-blind people can use. Given these constraints we give a list of properties that our user interface must possess.

Characteristics of Deaf-Blind Users

The main characteristic of deaf-blind people that impacts a user interface design is that they must, by necessity, receive information slowly. Those who are totally blind must receive information tactually and those with limited vision must read slowly. About fifty percent of deaf-blind people have a genetic disorder called Usher's syndrome which causes deafness at birth and a gradual degeneration of the retina during life. A person with Usher's syndrome will, from his or her teenage years, have tunnel vision, with the field of vision reducing through the years until full blindness is reached, typically in the person's thirties or forties. A person with severe tunnel vision, less than ten degrees of field of view, cannot view a typical CRT display all at once. The display must be scanned to pick up all the information displayed.

Many deaf-blind people do not have a good command of the English language. Those with Usher's syndrome are usually fluent in American Sign Language (ASL). Remarkably, ASL can be understood reasonably well tactually so that, once an Usher's syndrome person loses his or her sight, he or she can still converse in ASL. Special care must be taken in the design of any user interface and accompanying documentation to take into account the wide variation of educational backgrounds and English skills of its users.

Input/Output

In general, blind people can master the use of keyboards equipped with tactual indicators on the keys. Many deaf-blind people, particularly those who have Usher's syndrome, are familiar with typical keyboards because they have used typewriters and TDD's (telecommunication devices for the deaf). As a consequence, keyboard input to the computer is not a serious problem.

The physical problem of receiving information from the computer can be solved in a number of ways. For those who are totally deaf but have enough residual vision to read large print, there are commercially available large font displays. For those who are totally blind but with enough residual hearing to understand speech, there are a number of "talking" programs which use speech synthesis hardware. For the deaf-blind person who is both totally deaf and totally blind there are paperless Braille displays. A paperless Braille display is a device for producing Braille which uses small pins which can be raised or lowered to form Braille characters. Typical paperless Braille displays are one or two lines of twenty characters each. There are other tactual output devices that have been built. A simple Morse code output device has been built from an audio speaker because its vibrations can be felt [6]. An experimental finger-spelling hand has been developed at the Smith-Kettlewell Institute. A deaf-blind person can feel the shape of the finger-spelling hand to receive characters. An important tactual device is the Optacon, which permits the user to read either printed material or certain CRT monitors. It converts the printed symbol into a raised image representation on an array of vibrating pins which can be read one symbol at a time. With an Optacon, a blind user can read a standard visual display of text enabling interaction with standard textually-oriented software. Unfortunately, the Optacon is fairly expensive and requires a high degree of tactual sensitivity. There are very few deaf-blind people who use Optacons.

The output devices described above are all low bandwidth, which is not a severe limitation for their intended users; deaf-blind users only receive information slowly anyway. There is some potential for higher bandwidth output devices like the full-page tactual device being developed at the American Foundation for the Blind. Such a device would allow a full page of Braille to be displayed at one time and also allow some degree of graphical display. However, our interface is to directed toward currently available output devices which match the capabilities of deaf-blind users.

Properties of the User Interface

Naturally, there are basic considerations that go into the design of any user interface [3]. We take those considerations, such as ease of use, ease of learning, speed, accuracy, and error-avoidance, for granted. In addition, we are faced with additional considerations that are forced upon us due to our users' capabilities and the limitations of the output devices that are suitable for them. Because of these special considerations we have made a number of decisions which require explanation.

We chose to concentrate our efforts on output devices that produce either *large-font* or *Braille*. Our intended users already read large-print or Braille text. Those with Usher's syndrome are now learning or intend to learn Braille when they completely lose their vision. Braille is a textual medium - graphical images cannot be represented on a twenty to forty character Braille display. Since many of our users do not possess strong English comprehension skills, we could have attempted to use non-English icons or other graphical techniques with our users who have low vision. Because many, if not most, of our low vision users will eventually go fully blind we chose to make our visual and Braille display outputs as similar as possible. Since Braille is a representation of text, we are forced to use text in the form of large-font on our visual display. An attempt is made to restrict the vocabulary of the system to one that is likely to be well understood by our users.

Because our users are, in general, computer novices, we decided to make our system as *simple* and *uniform* as possible. Simple means that there are very few things to memorize to be able to use the system. The conceptual model behind the user interface is simple: menus help guide the user through the system, and there are ways of getting added assistance when needed. There are a number of different applications in the system (electronic mail, electronic news, and such), all of which are accessed in the same uniform way. When new applications are added in the future they will adhere to the same model. Simplicity and uniformity will help insure that the system is easy to learn and use.

For example, the Large Print Display Processor is manufactured by Vtek (formerly VISUALTEK), Santa Monica, CA.

For example, the Enable Reader Professional Speech System by the Enable Software Company, Berkeley, California.

The Braille Display Processor is manufactured by Vtek. The Microbrailler is manufactured by Triformation Systems, Inc., Stuart, Florida. The VersaBraille System is manufactured by Telesensory Systems Inc.

Optacon is manufactured by Telesensory Systems Inc.

We decided to make the user interface as *terse* as possible. Although reading is necessary, we try to minimize it. One means we chose to achieve terseness was to use a hierarchical system of menus [4]. We employ a single hierarchy for both commands and data. This allows the user to accomplish a task by making only a small number of choices along a path in the hierarchy.

For those deaf-blind people who rely on tactual output there is an additional problem. They generally receive information through the part of their body with the most tactual acuity: their hands. For example, a Braille user reads with his or her hands. This means that the Braille user cannot be reading the typing at the same time as doing the typing. The hands must be continually moving back and forth from the keyboard to the Braille display. With the hands doing both input and output, special care must be taken to allow the user to correct mistakes. Therefore, we made the decision that whenever input is requested from the user, he or she has full editing capability. We call this property the *pervasive editing capability.*

One critical decision we made was to develop a user interface which directly provides the functionality we were looking for. We realize that there are existing services that one can access which already provide many communication services. We could have tried to build a user interface which provides a way of viewing the standard CRT display of existing software. We call this the *window approach.* In the window approach, the large font or Braille output device, governed by appropriate software, gives access to a small part, or window, of a CRT. Regardless of the size of the window it must still be scanned by the user one character at a time. A large window can be a disadvantage, for it might force the user to hunt through the whole window to find what he or she is looking for. If the window positioning software is "smart enough" it can find what the user is looking for, then display it on the small window. Thus, the window approach provides a user interface into another user interface. We decided not to use the window approach because:

- The existing communication services are too verbose. They are tedious to access at 300 Baud and would be even more tedious at the bandwidths our users require.
- In the window approach our users would be forced to learn one user interface to access another. We felt this would be too complicated.
- Functions such as node-to-node talk and node-to-TDD talk do not exist in the existing services.

Instead of using the window approach, we chose to build special software to directly provide the functions we want. We call this the *custom software approach.* Our users will only have to learn our user interface. Naturally, there will have to be a way for the DBNet system to get information, like news, from existing services, but this will not be done by our users directly. The DBNet system will present the information it has acquired to the user in its own way. The custom software approach has advantages and disadvantages over the window approach. The main disadvantage is that it takes more effort to develop. The main advantage is that it is more likely to become useful because it better takes into account the limitations of its users and its intended output devices.

For example, CompuServe provides electronic mail and news.

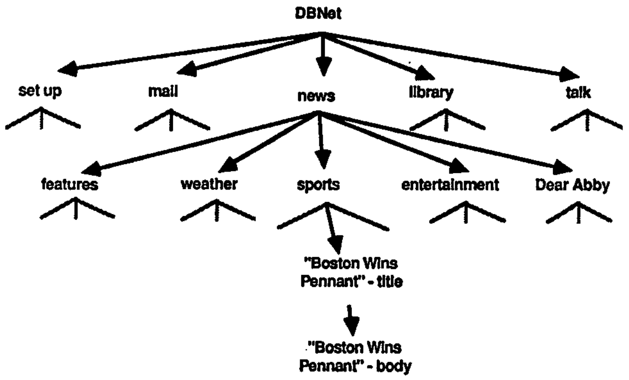

Figure 1: The Hierarchy of the DBNet system with "news" expanded.

3 DBNet User Interface

With the properties of our user interface in mind, we begin a description of DBNet. We will begin with a conceptual model of the system, then describe the physical actions the user must take to make the conceptual model a reality.

We make a distinction between a *task* and an *action.* A task is something to be accomplished by the user, such as reading the newspaper, or sending a letter. A task may have some subtasks as well, but we call only the most primitive tasks actions. Thus an action is usually accomplished by entering a single key stroke (usually a function key) or by entering a small amount of text (requiring several character key strokes). A task is accomplished by a sequence of actions.

The Model

The model of the system is a general *list*, that is, a hierarchical list with entries that can themselves be lists. Each list entry has *text* associated with it. The text may represent:

- A menu item to be selected by the user.
- Text to be read by the user.
- Text to be input by the user.
- Another kind of action to be taken by the user.

Such a list of lists can be represented by a tree in a natural way. Figure 1 shows the hierarchy of the DBNet system. The children of a node in the tree always represent distinct choices that lead to the accomplishment of a task. For example, the children of "news" in the tree of Figure 1 are "features", "weather", "sports", etc., which are the different categories of news that can be chosen from. The children of "sports" are the titles of sports articles. The child of a sports title is the body of that article itself. A path from the root to a node, in general, represents the actions and data that are needed to accomplish a task.

Both control and data are organized in the same way in the hierarchy. For example, in Figure 1 the "news" node represents control, a choice that the user might make among "set up", "mail", etc. The "Boston Wins Pennant - body" node represents data, the actual article to be read by the user. Generally, the organization is dynamic - lists can change while the system is in use. For example, the list of sports articles would change from day to day. In general, one node in the tree is the

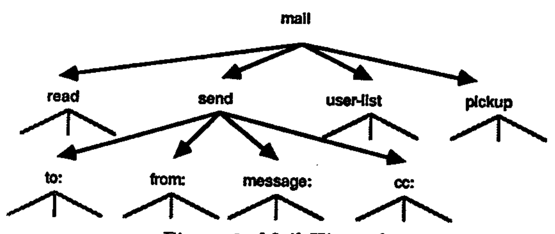

Figure 2: Mail Hierarchy

"current node," the node currently being visited by the user. While the user is at a node, the text of the node can be read and an action can be taken. An action taken by the user may allow him or her to:

- Read more of the text at the node if the text is longer than the available display.
- Write text at the node if input is required.
- Go to a sibling.
- Go to the parent.
- Go to a child.
- Do some other action which causes a specific effect such as changing font size, deleting a mail message, or placing a book mark.

Tasks to be accomplished by the user would be composed of sequences of these actions. The actions can be characterized simply by saying that they allow the user to traverse the tree, read and write the text at nodes of the tree, and effect other changes.

The hierarchical list was chosen as the basic model for a number of reasons. First, it is a natural organization for both action and data. Second, it is a way of limiting the number of choices to be made at one time to just those choices that appear as siblings in the tree. Finally, in just a few levels of a very limited branching tree thousands of complete tasks can be represented. The desired task can be performed by taking just a few actions. For example, the user can get to the sports articles, by first selecting "news", then selecting "sports" within "news". This is akin to first picking up the newspaper and then going straight to the sports section. The deaf-blind user can get to the desired task without time-consuming browsing or memorization of cryptic commands.

One possible disadvantage of a hierarchy is that there might be more than one sequence of actions that leads to accomplishing the same task. For example, in preparing a mail message one can prepare the body first, then address it second or vice versa. In such cases we attempt to avoid making one task a subtask of the other. We would make such tasks siblings in the hierarchy as seen in Figure 2. If one task is an immediate subtask of another then we would attempt to make the first a child of the second in the hierarchy. For example, preparing the body of a mail message is a subtask of writing the message as exemplified in Figure 2.

Realization of the Model

What must the user do to manipulate the model described above? When the user is at a node, the text associated with the node is normally presented on the display, either Braille or large font. In many cases the text associated with the node is longer than the available display length so only part of the text can be displayed. If there is more text to be seen, then the user is informed of that by the display of a special *more-to-scan* character at the right edge of the display. In some cases the user is requested to provide input. In those cases the system goes into *cursor active mode* and a cursor is displayed at a character position. For the large font display this is done by the inversion of the "pixels" at the position. However, this technique will not work with the Braille display because the inversion of the dots of a Braille character can represent another Braille character. For the Braille display the cursor is represented by the repeated reversal of the dots of the Braille character under the cursor. This creates a blinking effect, the rate of which is adjustable by the user.

There is a very small set of primitive actions that enable the user to have complete control of the DBNet system. These actions are performed by pressing appropriate function keys. The actions the user can perform can be categorized as follows:

Reading: In cases where the text at a node is longer than the length of the display, two function keys provide scaning capabilities. The first scans forward one displayful and the second scans backward one displayful. We call these keys *FORWARD* and *BACKWARD* respectively.

Writing: If input is requested from the user then cursor active mode is entered. In cursor active mode a set of simple text editing actions is enabled. Text may be entered at this point using the character keys on the keyboard. When a character key is typed the corresponding character is inserted at the cursor and the text following it shifts one position to the right. The *DELETE* key is also active, which allows the user to delete the character just before the cursor. In addition, cursor movement keys become active which allow the cursor to be moved left or right one character position. These cursor movement keys are called *LEFT* and *RIGHT* respectively.

Stepping: Two function keys, *NEXT* and *PREVIOUS*, are provided to allow the user to visit the siblings of a node. Generally, the siblings of a node are in a cyclical list.

Selecting: A function key *SELECT* is provided to allow the user to choose one of the siblings. Generally, the select key changes the current node to one of its children. In some cases depressing the *SELECT* key has a side effect.

Exiting: A function key *EXIT* is provided to allow the user to visit the parent of the current node.

There are a number of additional actions that provide shortcuts and assistance.

Example

The DBNet system can be configured in such a way that very little input is requested from the user. That is, the system is rarely in cursor active mode. Below we describe the use of the system when the cursor is not active. We demonstrate the DBNet system with the "news" application. The news application mimics a traditional newspaper. A newspaper is divided into a number of sections, for example, "features," "weather," "sports," etc. Each of these sections could consist of subsections or perhaps just articles. In our example, the "sports" section contains a list of sports article titles and under each title is the body of the article. Figure 1 shows the structure of news. Our example is intended to exemplify the general structure of the menus in our guided system, and to illustrate how the user makes choices.

In what follows, the text contained within a box indicates what would be displayed with a single-line display like a Braille display. The displayed text is part of the text at the current node. Text too long to fit within the box (i.e. display) can be read by using the scanning keys. If the text exceeds the length of the display, then the box contains the character "$" which means that there is more to scan to the right. When possible, text is interrupted only at word boundaries.

When the user starts up the system the first thing that is displayed is:

```
set up
```

This indicates that the current node is the sibling "set up" in the main DBNet list of applications. If the user presses the *NEXT* key then the current node becomes "mail". The display would now show:

```
mail
```

Pressing the *NEXT* key again will produce the display:

```
news
```

If the user wishes to see more choices then the *NEXT* key can be repeated. If the user wishes to read the news then *SELECT* is pushed while "news" is displayed. The display will now show the first category of news, "features":

```
features
```

The user can now repeatedly strike the *NEXT* key to find the other categories of news: "weather," "sports," and others. Once a category is decided upon, such as "sports," the user can choose to see the list of titles of sports articles by hitting the *SELECT* key when the display shows:

```
sports
```

At this point the title of the first sports article appears on the display. Again, by repeatedly hitting the *NEXT* key the user can review the various titles. If a title of a sports article is chosen by hitting the *SELECT* key then the body of that sports article is displayed, or at least as much of the beginning of the body as will fit on the small display. In other words, the first "displayful" is presented to the user. Suppose that the sports article chosen was titled "Boston Wins Pennant," then the text of the article might begin with:

> 9/28/86 AP - After leading the American League East for most of the season, the Boston Red Sox clinched the pennant. ...

The initial display when this article was chosen would be:

```
9/28/86 AP - $
```

The "$" symbol displayed means there is more to scan. The user could proceed to read the document by scanning the text. During scanning, whenever possible, words are not divided between two displays. Pressing the *FORWARD* key will scan forward:

```
After leading $
```

Pressing the *FORWARD* key again would result in:

```
the American $
```

Repeated pressing of the *FORWARD* key would allow the user to completely read the article. If the user wishes to go back one displayful he or she need only press the *BACKWARD* key. Once the user is finished reading the body of "Boston Wins Pennant", he or she can press *EXIT*. At that point the title of the sports article is the current node with the text of the title displayed.

```
Boston Wins $
```

At this point the user can examine more sports article titles by pressing *NEXT* or close the sports section by pressing *EXIT*.

Other Features

Additional features of DBNet provide assistance or shortcuts, making the system easier to use.

Two special function keys, *HELP* and *WHERE-AM-I?*, extend assistance. When the *HELP* key is depressed text explaining the purpose of the current node is displayed. When the *WHERE-AM-I?* key is pressed then a textual representation of the path from the root to the current node is displayed. For example, if the user is currently reading the body of the sports article "Boston Wins Pennant" then pressing *WHERE-AM-I?* would yield the text:

> "news: sports: Boston Wins Pennant: reading:"

which describes the path to the current node.

There is a "virtual bell" which "rings" if the user tries to perform an action that is not possible, such as scanning forward when there is no more text to scan. We call it a *virtual* bell because it has the function of the bell in a system that would be used by hearing users in alerting them to something. On the Braille display the virtual bell is "rung" by having all the pins on the display repeatedly go up and down several times. On the large font display the screen is reversed repeatedly several times. If the virtual bell rings then, by simultaneously pressing the "Alt" and *HELP* keys, the user obtains a message explaining why the bell rang.

Any assistance message can be scanned normally using the scanning keys, *FORWARD* and *BACKWARD*. The user leaves the assistance message and returns to his work by pressing the *EXIT* key.

Several of our basic actions also have a "strong version", which can be actuated by simultaneously pressing the basic key involved and the "Alt" key. For example, if the user presses the "Alt" and *EXIT* keys simultaneously then that is equivalent to pressing *EXIT* as many time as needed to get to the top level in the system. The keys *EXIT*, *FORWARD*, *BACKWARD*, *LEFT*, *RIGHT*, and *DELETE* all have strong versions.

Most novice users of DBNet will normally step through a list of alternatives, then select one, repeatedly until the task he or she is attempting to accomplish is reached. An advanced user may want a quicker way of moving around the system. The system can be put in *search mode*, which allows the user to jump around in the system directly without stepping or selecting. In search mode, if the user is at an appropriate node he or she can enter a path name relative to the current node. If a unique prefix of a path name is entered then the user will be presented with the complete path name and asked to confirm it before moving to that node. Search mode is not yet fully implemented.

4 Experience with DBNet

In April of 1986 we completed a "test version" of the DBNet system and began testing the system with a variety of deaf-blind users. The purpose of the test was to determine if the deaf-blind individuals could learn the conceptual model behind the system and successfully manipulate it to perform tasks. We wanted to know, before we went on to a full version of the system, whether or not it could be used and how easy it was to use. We wanted to know what aspects of the system should be changed to accommodate deaf-blind users. It was not our intention to do a thorough scientific study. Such a study would be very difficult to set up because the sample group is so small and diverse, and because there is practically no other system to compare to DBNet.

We selected six deaf-blind users in the Seattle area and taught them how to use the system on an individual basis. Each initial teaching session was approximately two hours. The teaching was done in the users' homes and some of the users were allowed to keep the machine for a few days to familiarize themselves more with the system. Approximately two weeks after the initial instruction we brought the system back to the users to test them on their comprehension and retention of the system. The test basically asked them to perform a number of tasks, such as "find and read the sports article about Boston winning the pennant" or "change the font to a smaller size". We simply observed how easily the users were able to accomplish these tasks.

The teaching was done by several different teachers, most often through a tactual sign language interpreter. Since the subjects had diverse educational levels, language skills, and deaf-blindness, different teaching styles were employed. During the first few teaching sessions we discovered several problems with the system. For example, the application "set up" was initially called "profile". This caused some confusion because the word "profile" was not a familiar word to the users.

The most interesting question for us was: *can this group learn the conceptual model underlying the DBNet system and can they learn the actions to successfully manipulate that model to get things done?* It is our opinion that five out of six of the subjects successfully understood the underlying model and learned the actions to manipulate it. The five that successfully learned the system in one teaching session seemed to grasp the model in less than an hour. In those cases the second hour was used for practice.

Since our initial tests on the six subjects, approximately twenty-five other deaf-blind users have learned the system to some degree. We brought the system to the American Association of the Deaf-Blind (AADB) Convention in Washington D.C. in June 1986 and to the Deaf-Blind Camp sponsored by the Seattle Lighthouse for the Blind in August 1986. It is our estimate that twenty out of the twenty-five were able to learn the system with less than one hour of teaching. In some cases, as little as ten minutes of teaching was needed.

5 Conclusion

We have designed, built, and tested a software system suitable for deaf-blind users who read output with either a large font or Braille display. The user interface is based on an extremely simple hierarchical model which allows its users to perform hundreds of distinct tasks by taking just a few actions. The hierarchical model is well within the conceptual grasp of deaf-blind people and can be learned with very little teaching effort.

We are now in the process of implementing the rest of DBNet. We are implementing full-fledged electronic mail and news systems. We are confident that the user interface that we have designed will make it possible for the majority of deaf-blind people to learn and use the full functionality of DBNet.

ACKNOWLEDGMENTS. We would like to thank Kathy Briscoe, Ann Sauer, and Dave Socha for helpful comments and criticisms of early drafts.

References

[1] Roxanne Starr Hiltz and Murray Turoff. *The Network Nation, Human Communication via Computer.* Addison-Wesley Publishing, 1978.

[2] Richard E. Ladner and Barbara J. Wagreich. Networks for deaf-blind people. In *Proceedings of the Compcon '84*, pages pages 341–344, IEEE Computer Society Press, 1984.

[3] Marilyn Mehlmann. *When People Use Computers: An Approach to Developing an Interface.* Prentice-Hall, Englewood Cliffs, New Jersey 07632, 1981.

[4] Ricky E. Savage and James K. Habinek. A multilevel menu-driven user interface: design and evaluation through simulation. In John C. Thomas and Michael L. Schneider, editors, *Human Factors in Computer Systems*, chapter 7, pages 165–186, Ablex Publishing Corporation, Norwood, New Jersey 07648, 1984.

[5] Barbara Wagreich. Electronic mail for the hearing impaired and its potential for other disabilities. *IEEE Transactions on Communications*, 30(1):pages 58–65, 1982.

[6] Dan Zuckerman. Use of personal computing technology by deaf-blind individuals. *Journal of Medical Systems*, 8(5):431–436, 1984.