

# Towards Accessible Human-Computer Interaction<sup>1</sup>

**Eric Bergman**

*SunSoft*

**Earl Johnson**

*Sun Microsystems Laboratories*

## INTRODUCTION

In spite of the growing focus on user-centered interface design promoted by human-computer interaction researchers and practitioners, there remains a large and diverse user population that is generally overlooked: users with disabilities. There are compelling legal, economic, social, and moral arguments for providing users with disabilities access to information technologies. Although we will touch on some of those arguments here, the primary purpose of this chapter is to provide a broad overview of the software human-computer interaction (HCI) issues surrounding access to computer systems.

The needs of users with disabilities are typically not considered during software design and evaluation. Although there are many plausible explanations for this omission, we are inclined to believe that much of the problem simply stems from lack of awareness. Until recently, there was little contact between HCI organizations and the sizable disability community of people with disabilities (McMillan, 1992). As a result, many software designers are not aware of the needs of users with disabilities. We hope that this chapter will serve to increase awareness of the scope and nature of these needs, and to stimulate interest in research and implementation of systems that enable access to information technologies by users with disabilities.

Designing software that takes the needs of users with disabilities into account makes software more usable for all users: people with disabilities who use assistive technologies, those who use systems "off the shelf", as well as users without any significant disabilities. Considerable literature already exists that discusses how people with disabilities can use assistive hardware and software to interact with computers. For this reason, we provide relatively brief coverage of assistive technologies here.<sup>2</sup>

In this chapter, we define and discuss accessibility, discuss accessibility design issues, provide a broad outline of the capabilities and needs of users with various disabilities, present guidelines for increasing application accessibility, and discuss future

- 
1. The authors would like to thank Ellen Isaacs, Elizabeth Mynatt, Mark Novak, and Will Walker for their valuable suggestions and comments on this chapter
  2. For more detailed information on assistive technologies see Lazzaro (1993), Church and Glennen (1992), Brown (1992), and Casali and Williges (1990).

directions for improving accessibility.

## THE RELEVANCE OF ACCESSIBILITY

### What is Accessibility?

Providing accessibility means removing barriers that prevent people with disabilities from participating in substantial life activities, including the use of services, products, and information. We see and use a multitude of access-related technologies in everyday life, many of which we may not recognize as disability related when we encounter them. The bell that chimes when an elevator is about to arrive, for example, was designed with blind people in mind (Edwards, Edwards, and Mynatt, 1992). The curb cut ramps common on street corners in the United States were introduced for wheelchair users (Vanderheiden, 1983). Accessibility is by definition a category of usability: software that is not accessible to a particular user is not usable by that person. As with any usability measure, accessibility is necessarily defined relative to user task requirements and needs. For example, a telephone booth is accessible (e.g., usable) to a blind person, but may not be accessible to a person using a wheelchair. Graphical user interfaces are not very accessible to blind users, but relatively accessible to deaf users.

Vanderheiden (1991) makes a distinction between “direct” access and access through add-on assistive technologies. He describes direct access as “adaptations to product designs that can significantly increase their accessibility...” (p. 2). A major advantage of this approach is that large numbers of users with mild to moderate disabilities can use systems without any modification. Examples of direct access include software keyboard enhancements included with X Windows, OS/2, and the Macintosh (see Table 2).

Assistive access means that system infrastructure allows add-on assistive software to transparently provide specialized input and output capabilities<sup>1</sup>. For example, screen readers (see Table 3) allow blind users to navigate through applications, determine the state of controls, and read text via text to speech conversion. On-screen keyboards replace physical keyboards, and head-mounted pointers replace mice. These are only a few of the assistive technologies users may add on to their systems.

We claim that in order to truly serve users with disabilities, accessibility must mean more than simply providing “direct” access through assistive technologies bundled with system software, and more than providing the capability to add such assistive technologies. It also must mean designing application user interfaces that are easier to use for users with disabilities as well as users “with out” disabilities by taking their needs into account when system and application software is designed.

---

1. By *infrastructure* we mean an environment’s standard set of application program interfaces (APIs). These are low and high-level software routines used to build applications (e.g., Macintosh Toolbox, MS Windows API, Motif, and Xlib to name a few).

## **Broad Benefits of Accessibility**

Accessibility provides benefits for a wide range of people -- not only for those with disabilities. Before curb cut ramps were placed on sidewalks, for example, it was difficult or impossible for people in wheelchairs to cross a street. In addition, curb cuts have benefited people pushing baby carriages and shopping carts as well as those on bicycles and roller blades. Vanderheiden (1983) suggested that our society is laying down electronic equivalents to sidewalks. These "electronic pathways", he argued, must include electronic "curb cuts".

Like physical curb cuts, electronic curb cuts provide benefits for the larger population as well as users with disabilities. Systems that allow use of keyboard shortcuts instead of the mouse increase efficiency of sighted users as well as providing access for blind users or users who have disabilities that affect mouse control. Users of portable computers or people in open offices may not be able to use or hear sounds, but they can use visual cues, as can hearing impaired users. Users who must keep their eyes on their task (e.g., aircraft mechanics) can benefit from systems that interact through voice rather than vision, as can users with visual disabilities.

In the near future, when the power of computing is available on noisy factory floors, in cars hurtling down expressways, and from devices stuffed in our pockets, the relevance of accessibility looms larger. As users and designers, we will soon deal with environment and task demands in which systems must deliver computing power to people who will be unable to use their hands, eyes, or ears to interact with computers. People working on accessibility have been tackling such design issues for years, typically with little or no input from HCI specialists. Clearly there needs to be a better communication between the disability access and HCI communities.

There are ample incentives for fostering a connection between design for access and design for "general audiences". Research on communication devices for the deaf led to development of the telephone, whereas development of an easy to use "talking book" for the blind led to the cassette tape (Edwards, Edwards, et al., 1993). HCI theory and practice can benefit from better understanding of the difficult issues that are already being addressed in the disability domain. Newell and Cairns (1993) cite designers who thought they had created novel interfaces, which were actually reinventions of disability access technologies such as foot-operated mice and predictive dictionaries<sup>1</sup>. As McMillan (1992) suggested, cooperation among these different communities will require better communication among professionals in the fields of rehabilitation education and HCI.

## **Economic and Social Impact**

A common argument is that computer accessibility is too costly, but in reality inaccessible systems may cost much more. Government statistics show that there is a growing market for accessible computer products. Approximately 43 million Americans have a dis-

---

1. Predictive dictionaries speed typing by predicting words as the user types them, and offering those words in a list for the user to choose. Originally intended for users with movement related disabilities, predictive dictionaries are now becoming popular for users with RSI and as a way to boost typing speed.

ability of some type (Perritt, 1991). According to Elkind (1990), about 15% of the population has a disability “severe enough to interfere with work or otherwise constitute a serious handicap”, whereas Vanderheiden (1990) points out that over 30% of the people with disabilities who want to work are not employed.

There are human costs that are harder to measure than numbers from tax tables or percentages of people employed. As a user with low-vision told us, "I don't believe in telling peers or management [that] I can't read it on the computer so I can't do my work". We have spoken to blind users who, because they are unable to read text that they have typed into word processing applications, have secretaries or coworkers type for them. We have spoken to users with mobility impairments who could not use job-critical applications because they were not accessible from the keyboard. Our experience suggests that a significant proportion of users with disabilities who are employed are not working at full productivity due to poor software accessibility.

### **Legal Requirements**

In recent years, new laws have created incentives for the development of access to computer technology (see Casali & Williges, 1990; Lazzaro, 1993; McCormick, 1994). The most important of these laws, for the purposes of our discussion, are Section 508 of the 1986 Federal Rehabilitation Act and the 1992 Americans with Disabilities Act (ADA).

#### **Key aspects of Section 508:**

- Applies to office computer purchases made by the Federal Government
- Winning contract proposals must include solutions for employees with disabilities

#### **Key aspects of the ADA:**

- Applies to corporate entities with 15 or more employees
- Requires "reasonable accommodation" for employees with disabilities

One of the driving forces behind these laws has been the government's economic and social interest in retaining and recruiting people with disabilities. In fact, the government employs over 151,000 people with disabilities (McCarthy, 1994). The ADA was seen as a way to make the private sector a partner in hiring and retaining employees with disabilities, reducing demand for government assistance. The growing market for accessible technologies means that accessibility can be an important selling point for system and application software in both commercial and federal markets.

## **DESIGNING FOR DISABILITIES**

Arguments typically leveled against designing for users with disabilities include the claims that costs are too high, and the benefits serve too small a market (Glinert & York, 1992). These arguments should sound familiar to HCI practitioners, who have historically faced initial resistance or even opposition to the introduction of HCI processes into product

development. Just as organizational understanding of the design process must be changed for HCI to be accepted, so too must the standard HCI conceptualization of "the user" change to recognize the needs of people with disabilities.

### **Ranges of User Capabilities**

The traditional view of people "having a disability" or "not having a disability" is overly simplistic. All users have a range of capabilities that varies across many dimensions depending on the user and his or her life stage, task, and environment. As more of the population approaches their middle 40's, there are increasing numbers of users peering through bifocals at their screens. A nontrivial portion of the population experiences some degree of hearing loss, and may not always notice software alert sounds. As we age, more of us will develop age related disabilities -- 25% by age 55, jumping to 50% at age 65 (Vanderheiden, 1990).

In addition to normal consequences of aging, people may experience sudden temporary or permanent changes in capabilities at any time in their lives. If a computer user falls and breaks her wrist, she will spend several weeks or more with much the same keyboard capabilities as many people with spinal cord injuries or missing limbs. Every user's ability to interact with systems varies over short as well as long periods of time. A sprained wrist can restrict use of a mouse. A late night working or a lost contact lens can transform small fonts into a suddenly challenging experience. Any user who does not currently have a disability could someday have a stroke, car accident, or other event resulting in a temporary or permanent disability.

In fact, a significant number of user requirements for people with disabilities apply to almost any user, given the right circumstance or task context (Edwards, Edwards, et al, 1993; Newell and Cairns, 1993; Edwards, et al, 1993). Whether a user's hand is broken, painful due to repetitive strain injury, or permanently paralyzed, there are similar needs. Whether someone is unable to look at a screen because he is driving, or cannot see a screen because he is blind, the user requirements have much in common. Whether a user does not hear because she is talking to somebody on the phone, paying attention to her task, working in a noisy environment, or happens to be deaf is less important than the fact that users in these contexts need alternate sources of information. As McMillan (1992) observed, "From the point of view of a computer, all human users are handicapped" (p. 144).

### **Who is "the User?"**

Who is "the user" -- that familiar catchphrase we encounter in papers, conference sessions, and design sessions? Does it include users with disabilities?

Does "the user" include a programmer we visited? She was diagnosed with muscular dystrophy in her early 20's. This condition, which results in progressive loss of muscular strength, means that she works from her motorized wheelchair, and is unable to sit upright for more than a brief time. As a result, she works in a reclined position, leaning back almost horizontally. Her vision problems limit the amount of time she can focus on the screen, and her muscular weakness prevents her from handling paper manuals.

Does "the user" include a secretary we interviewed? She has no vision in one eye and

"tunnel vision" in the other and prepares documents using a standard PC and screen magnification software. Sometimes she is unable to tell the difference between old and new email messages, because her mail application uses color to distinguish old from new. Like many users with low vision, she has problems working with columns, because it is difficult for her to see if text is aligned.

Does "the user" include a writer we know who took several months off from work when she developed tendonitis? She was not able to type or use a mouse for more than a few minutes, nor was she able to lift heavy objects, including manuals. Although her condition improved significantly over time, for several months she had a disability which affected her work significantly.

Does "the user" include a computer support technician we met? He has cerebral palsy, and is able to use only his left hand. There are keyboard assistive applications that would help speed his typing, but his work requires moving from computer to computer installing and troubleshooting Microsoft Windows software, which, at this writing, has no *built-in* access features<sup>1</sup>. It is not practical for this user to install an assistive keyboard application for each consultation, so he instead stretches his single usable hand wide to reach and press multiple keys when he needs control keys and capital letters. Fortunately for him, and his employer, he has a large hand.

These users and many others have told us that their needs are not being met by current computer systems. Users with physical disabilities complain about applications that cannot be controlled from the keyboard. Users with low vision describe software that does not allow them to adjust the color to make text legible. Blind users complain about documentation that is not accessible because it is not available on paper in braille or on the computer as plain text (which is required for screen reading applications).

Access problems are not confined to users who have a "classic" disability. As they age, users who would claim they have no disability find that screens become more difficult to read and sounds become more difficult to hear. Users who break an arm, sprain a wrist, lose a contact lens, require bifocals, or develop repetitive stress injuries suddenly find that computer systems do not take their needs into account.

### **Limited View of the User**

A common argument goes something to the effect that software engineers typically design for themselves, while HCI professionals follow a process based on understanding user characteristics, needs, tasks, and environments. In spite of this claim to the domain of user needs, we argue that most HCI research, literature, and practice holds forth a relatively limited view of who constitutes "the user." Most of the people we have discussed as examples of users with disabilities would not fit in this limited view.

Nielsen's (1993) Usability Engineering, for example, discussed the user in a section titled "Categories of Users and Individual Differences." Nielsen focuses on user experi-

---

1. Note that this is scheduled to change in "Windows 95". As demonstrated in this case, however, such add-on software alone does not meet the needs of a mobile user.

ence with computers in general, the system in particular, and the task at hand. He noted that "users also differ in other ways than experience" and went on to list such attributes as age, gender, spatial memory, and learning style. Users with disabilities are not mentioned, in spite of the explicit focus on "Categories of Users." In fact, disabilities are only mentioned in a few brief sentences in the entire book.

We use Nielsen's book only as an example of the extent to which disability issues are ignored by mainstream literature. Although there is a growing body of work on accessibility within the HCI community, it has not been generally recognized in standard texts or in work that is not explicitly focused on disability issues. Users with disabilities are simply not "on the radar screen" of mainstream HCI.

## **Universal Design**

In recent years, as the notion of accessibility has been broadened to encompass much more than design for people with disabilities, the concept of "universal design" has gained visibility. Traditional design has focused on filling the needs of the "average" person, with the assumption that design for the average provides for the needs of most. The universal design argument is that designing for the "average" is by definition exclusionary, because the "average" user is a fictitious construct.

Attempts to design for this fictitious "average" user may not account for effects of multiple individual differences. Tognazzini (1992, p. 74) related an anecdote from Blake (1985) illustrating the pitfalls of ignoring such overlapping differences:

Several years ago, the Air Force carried out a little test to find out how many cadets could fit into what were statistically the average-size clothes. They assembled 680 cadets in a courtyard and slowly called off the average sizes--plus or minus one standard deviation--of various items, such as shoes, pants, and shirts. Any cadet that was not in the average range for a given item was asked to leave the courtyard. By the time they finished with the fifth item, there were only two cadets left; by the sixth, all but one had been eliminated.

The Universal Design philosophy emerges from a recognition of the idea central to this story -- that there is no average user. Universal designs target the broadest possible range of user capabilities. Examples of products that embody this theme include automatic doors, remote control thermostats, and velcro. Using no assistive technology, people who were previously unable to open a door, operate a thermostat, or tie their shoes are able to perform these tasks, whereas "the rest of us" find these tasks easier as well. Proponents of universal design do not assume that all users will be able to use all designs, but instead argue that by redefining our definition of the user, a much wider range of users can be accommodated without significant extra effort (Vanderheiden, 1992a).

## **Watch out for that Ramp**

We claim that Universal Design is a worthy goal, but it is important to acknowledge that there are complex customization-related HCI issues that must be resolved before it can be achieved with computers. In discussing user interface design, Lewis and Rieman (1994) wrote, "Experience shows that many ideas that are supposed to be good for everybody aren't good for anybody." We agree that in human-computer interaction, as in much of life, what is "good for you" is not always "good for me."

An example of this principle in action was illustrated to us by a colleague who caught her foot at the base of a wheelchair ramp and tripped. The resulting fall produced injuries that included a sprained wrist and numbness in one hand. The injuries could easily have been more severe. The irony of a person acquiring a temporary or perhaps permanent disability because of an artifact designed to help people with disabilities strikes us as an appropriate alert that there may be stumbling blocks on the path to Universal Design.

One computer-related stumbling block is apparent in considering a simplified scenario of a public information kiosk. If we assume blind users must have access, then it becomes important to provide voice and sound output. There may be a tone when a control has been activated, and voice output to provide information about graphics and video displayed on screen. If a deaf user steps up to the kiosk, she will need visual notifications such as closed captions and visual alerts as alternatives to voice and sound alerts. If a user with no significant disability steps up to the kiosk, how will it interact with her? Surely she will not wish to deal with a cacophony of voice descriptions, closed captions, beeping dialogs and flashing screens?<sup>1</sup>

Environments are needed that allow users to tailor system input and output modalities to their capabilities and preferences. Recent research has suggested that information can be represented in a form abstracted from the particulars of its presentation (Blattner, Glinert, Jorge, and Ormsby, 1992; Fels, Shein, Chignell, and Milner, 1992). The technical solution of providing multiple redundant interface input and output mechanisms is not, in itself, sufficient to resolve conflicting user needs. In the absence of any means for intelligently designing and customizing their use, highly multimodal interfaces could lead to as many usability problems as they resolve, causing some users to trip over features designed to help other users. Determining how users will interact with such systems is a challenging HCI issue.

### **Include People with Disabilities in the Design Process**

It is in the design and evaluation of operating systems and desktop environments that designing for people with disabilities is most critical. Without the appropriate system software infrastructure, no amount of effort on the part of application developers can improve the accessibility of applications. As we discuss later, large gains in the accessibility of computer systems ultimately depend on improvements in software infrastructure.

On the other hand, there are many ways to improve the accessibility of applications within the constraints of current systems. Perhaps the most obvious way to enhance accessibility is to consider the needs of people with disabilities in all stages of the design process, including requirements gathering, task analyses, usability tests, and design guidelines. Other strategies include evaluating the usability of software in conjunction with popular assistive technologies, and testing under simulated disability conditions (e.g., unplug the mouse, turn off the sound, and use a screen reader with the monitor turned off).

---

1. This scenario is of course an oversimplification. Future systems are likely to adapt themselves to work intelligently with each user -- perhaps based on a stored profile. How such systems become configured and interact with users remains an interesting HCI question.

Note that none of these approaches are substitutes for testing with users. Simulation does not realistically represent the rich contexts and needs of users with disabilities. On the other hand, it is better than not testing accessibility at all.

Usability testing with even one user from each of the general disability categories we discuss in this chapter can have significant benefits for all users, not only those with disabilities. Depending on their disability, users can be especially affected by usability defects. Low vision users are sensitive to font and color conflicts, as well as problems with layout and context. Blind users are affected by poor interface flow, tab order, layout, and terminology. Users with physical disabilities affecting movement can be sensitive to tasks that require an excessive number of steps or wide range of movement. Usability testing with these users can uncover usability defects that are important in the larger population.

## **THE ROLE OF GUIDELINES AND APPLICATION PROGRAM INTERFACES**

### **Design Guidelines**

Although large leaps in accessibility await improvements in the design of operating systems and desktop environments, there is much that can be done to improve access within the constraints of current systems. The extent to which applications follow design guidelines, for example, can have a disproportionate affect on people with disabilities. User confusion about how to perform tasks is always a problem, but such problems become magnified for users who use alternative input and output devices or who require extra steps or time to navigate, activate controls, or read and enter text.

Keyboard mapping guidelines, for example, are especially important for those users who have movement impairments or are blind. Many of these users employ assistive software that assumes applications will use standard keyboard mnemonics and accelerators. Keyboard inconsistencies that are annoying to users without disabilities can become major roadblocks to users with disabilities.

Blind users of graphical user interfaces are especially affected by arbitrary violations of design guidelines with respect to application layout, behavior, and key mappings. These users interact with their systems through keyboard navigation, and use one-line braille displays and/or voice synthesis with screen reader software to read screen contents. Unlike a sighted user who can selectively scan and attend to screen elements in any order, blind users with screen readers move through a relatively linear presentation of screen layout. If that layout deviates significantly from guidelines, it can be especially difficult for a blind user to understand what is happening.

Because there are always exceptions, new situations, and missing guidelines, interface designers often have to violate guidelines or invent their own guidelines. In these cases, it is important that designers consider how any given violation might affect usability for users with disabilities. Unfortunately, most interface style guides were not written taking users with disabilities into account. For this reason, we have provided a set of general design

guidelines in this chapter that can be applied across a variety of environments.

### **Application Program Interface (API)**

Just as guidelines specify standard methods for interacting with an interface, various user interface application program interfaces (APIs) specify standard methods for applications to interact with each other and the system. APIs are the low and high-level software routines used to build applications. Every software environment provides standard API functions to support activities such as reading characters from the keyboard, tracking position of the pointer, and displaying information on the screen<sup>1</sup>.

Assistive software and hardware mediates the communication between users and applications, making it particularly sensitive to cases where standard API functions are not used. This sensitivity occurs because assistive software such as screen readers and speech recognition monitor the state and behavior of applications partly by tracking their use of API functions. For this reason, applications that do not use standard API calls have the potential to create serious usability problems for people with disabilities. If an application performs a common function (e.g., reading a character from the screen) without using standard API calls, assistive software may not know that an event has occurred, and consequently the user may not be able to use the application. For the same reason, a non-standard interface component (e.g., a custom control) can cause access problems because chances are good that assistive technologies will not be able to recognize it or interact with it.

The API problem is a complicated issue, and beyond the scope of this chapter to address in detail, but it is important that project teams discuss and weigh the trade-off inherent in departures from APIs. Reasons for not using standard API functions include:

- It is not possible to implement the desired functionality using a high level API alone, because it does not support required user interface features.
- The high-level API calls do not yield desired performance.
- Developers are more familiar with lower-level APIs that existed prior to the development of higher-level APIs. In this case, developers may be more efficient or comfortable implementing existing features using the features they already know.

There are no simple solutions to the API problem. Where possible, engineers need to use APIs above the level in which assistive software connects to the system. If using a standard API constrains application functionality or performance, the reality of product or task requirements often means that it cannot be used. In cases where developers are unfamiliar with high-level APIs, training may help. When an interface feature is not supported by an API, it is rarely practical to drop that feature. In cases where an interface has features that clearly circumvent basic API functions, every attempt should be made to insure that the

---

1. Common APIs include the Macintosh Toolbox, the MS-Windows API, and Motif.

tasks these features support can also be accomplished using other features. In the future, access problems related to API support can be reduced environments whose infrastructure provides built-in support for access.

## **ABOUT DISABILITIES: BACKGROUND AND DESIGN GUIDELINES**

In this section, we discuss some of the needs, capabilities, and assistive technologies used by people with disabilities, and we provide guidelines for improving application accessibility. The brief descriptions in this section do not constitute complete coverage of the wide range of disabilities, capabilities, needs, and individual differences across the population of people with disabilities--we have focused on providing a broad introduction to visual, hearing, and physical disabilities. Users with cognitive, language, and other disabilities may have needs in addition to those discussed in this chapter<sup>1</sup>.

Use of assistive technologies varies across users and tasks. Our discussion of assistive technologies is not comprehensive, but it does cover many commonly used software and hardware solutions. In reading this section it is important to remember that as with all users, the needs of users with disabilities vary significantly from person to person. Many users with disabilities do not use assistive technologies, but can benefit from small design changes. Other users have significant investments in assistive technologies, but they too can benefit from software that better responds to their interaction needs.

### **About Physical Disabilities**

Physical disabilities can be the result of congenital conditions, accidents, or excessive muscular strain. By the term "physical disability" we are referring to disabilities that affect the ability to move, manipulate objects, and interact with the physical world. Examples include spinal cord injuries, degenerative nerve diseases, stroke, and missing limbs. Repetitive stress injuries can result in physical disabilities, but because these injuries have a common root cause, we address that topic below under its own heading.

Many users with physical disabilities use computer systems without add-on assistive technologies. These users can especially benefit from small changes in interface accessibility. As a case in point, we recently met a manager with Cerebral Palsy who uses a standard PC and off-the-shelf Windows business productivity applications, but navigates almost exclusively via the keyboard because his poor fine motor coordination makes a mouse or trackball difficult to use. When a pointing device becomes necessary, he uses a trackball. As he explained it to us, "I hate the mouse".

Some users with physical disabilities use assistive technologies to aid their interactions (see Tables 1 and 2). Common hardware add-ons include alternative pointing devices such as head tracking systems and joysticks. The MouseKeys keyboard enhancement available for MS Windows, Macintosh, and X Windows-based workstations allows users to move

---

1. We believe that in many cases, consideration of the issues discussed here will address many of the needs of these users. See Vanderheiden (1992) and Brown (1989) for more information on other disabilities.

the mouse pointer by pressing keys on the numeric keypad, using other keys to substitute for mouse button presses. Because system-level alternatives are available, it is not necessary for *applications* to provide mouse substitutes of their own. The problem of the mouse is a good example of the kind of generic issue that must be addressed at the system rather than application level.

Unfortunately, the MouseKeys feature is often time-consuming in comparison to keyboard accelerators, because it provides relatively crude directional control. For tasks requiring drag and drop or continuous movement (e.g., drawing), MouseKeys is also inefficient. On the other hand, because current systems are designed with the implicit assumption that the user has a mouse or equivalent pointing device, many tasks require selecting an object or pressing a control for which there is no keyboard alternative. In these cases, MouseKeys provides an option. It is clear that future operating environments need to offer effective alternatives for users who may not use a pointing device.

It is important that applications provide *keyboard access* to controls, features, and information in environments that have keyboard navigation<sup>1</sup>. Comprehensive keyboard access helps users who cannot use a mouse. Many environments allow users to use tab and arrow keys to navigate among controls in a window, space bar and enter to activate controls, and key combinations to move focus across windows. In some cases, extra engineering may be required to ensure that these features work in all areas of an interface.

In addition to keyboard navigation, keyboard accelerators and mnemonics are also helpful for users with physical disabilities (as well as blind and low vision users). Whenever practical, commonly used actions and application dialogs should be accessible through buttons or keyboard accelerators. Unfortunately few of the standard accelerator sequences were designed with disabilities in mind. Many key combinations are difficult for users with limited dexterity (e.g., in Motif, holding down Alt-Shift-Tab to change to the previous window in Motif). Nonetheless, use of key mapping consistent with guidelines for the local application environment not only speeds use of applications for users with movement difficulties, but it also increases the effectiveness of alternate input technologies such as speech recognition. Assistive technologies often allow users to define macro sequences to accelerate common tasks. The more keyboard access an application provides, the greater the user's ability to customize assistive technology to work with that application.

### **About Repetitive Strain Injuries (RSI)**

Perhaps the fastest increasing disability in today's computerized workplace is repetitive strain injury (RSI). The Occupational and Health Safety Administration reported that 56 percent of all work place injuries reported during 1992 were due to RSI, up from 18 percent in 1981 (Furger, 1993). RSI is a cumulative trauma disorder that is caused by frequent and regular intervals of repetitive actions. Common repetitive stress injuries are tendonitis and carpal tunnel syndrome, although other types of injuries also occur. Symptoms of com-

---

1. The lack of keyboard control navigation in some user interface environments is an accessibility problem that needs to be addressed by the designers of these environments.

puter based RSI include headaches, radiating pain, numbness, tingling, and a reduction of hand function. For computer users, mouse movements and typing may be causes or contributors to RSI.<sup>1</sup>

Sauter, Schliefer, and Knutson (1991) found that repetitive use of the right hand among VDT data entry operators was a factor in causing RSI. They suggested that a change to “more dynamic tasks” could help reduce the likelihood of RSI. In general, users should be given the choice of performing a task using a variety of both mouse and keyboard options. For custom applications involving highly repetitive tasks, consider providing automatic notification for users to take breaks at regular intervals if there is no such capability at the system level.

Frequently repeated keyboard tasks should not require excessive reach or be nested deep in a menu hierarchy. We once met a customer support representative who had an RSI-related thumb injury. One of her most common jobs tasks -- changing screens to register information from phone calls -- encouraged an extremely wide stretch of her left thumb and forefinger in order to press a control and function key simultaneously. This thumb eventually required surgery.

The needs of users already having symptoms of RSI overlap significantly with the needs of users with other types of physical disabilities. Assistive technologies such as alternate pointing devices, predictive dictionaries, and speech recognition can benefit these users by saving them keystrokes, reducing or eliminating use of the mouse, and allowing different methods of interacting with the system.

**TABLE 1. Assistive Technologies for Physical Disabilities and RSI**

<b>Assistive Technology</b>	<b>Function Provided</b>
Alternate Pointing Device	Gives users with limited or no arm and hand fine motor control the ability to control mouse movements and functions. Examples include foot operated mice, head-mounted pointing devices and eye-tracking systems.
Screen Keyboard	On-screen keyboard which provides the keys and functions of a physical keyboard. On-screen keyboards are typically used in conjunction with alternate pointing devices.
Predictive Dictionary	Predictive dictionaries speed typing by predicting words as the user types them, and offering those words in a list for the user to choose.
Speech Recognition	Allows the user with limited or no arm and hand fine motor control to input text and/or control the user interface via speech.

---

1. We discuss software strategies for reducing RSI in this section. Note that there are a wide variety of commercial ergonomic keyboards and alternative input devices aimed at reducing RSI.

**TABLE 2. Keyboard Enhancements**

<b>Feature</b>	<b>Function Provided</b>
StickyKeys	Provides locking or latching of modifier keys (e.g., Shift, Control) so that they can be used without simultaneously pressing the keys. This allows single finger operation of multiple key combinations.
MouseKeys	An alternative to the mouse which provides keyboard control of cursor movement and mouse button functions.
RepeatKeys	Delays the onset of key repeat, allowing users with limited coordination time to release keys.
SlowKeys	Requires a key to be held down for a set period before keypress acceptance. This prevents users with limited coordination from accidentally pressing keys.
BounceKeys	Requires a delay between keystrokes before accepting the next keypress so users with tremors can prevent the system from accepting inadvertent keypresses.
ToggleKeys	Indicates locking key state with a tone when pressed, e.g., Caps Lock.

### **About Low Vision**

Users with low vision have a wide variety of visual capabilities. According to Vanderheiden (1992), there are approximately 9-10 million people with low vision. For the purposes of this chapter, consider a person with low vision to be someone who can only read print that is very large, magnified, or held very close.

We recently met a user who can read from a standard 21" monitor by using magnifying glasses and the largest available system fonts. His colleague down the hall must magnify text to a height of several inches high using hardware screen magnification equipment. A secretary we met has "tunnel vision", and can see only a very small portion of the world, as though she were "looking through a straw". In the region where she can see, her low acuity requires her to magnify text so that only a portion of her word processing interface is visible on-screen at any one time, reducing her view of the interface to only a single menu or control at a time. These are only a few of the wide variety of low vision conditions.

The common theme for low vision users is that it is challenging to read what is on the screen. All fonts, including those in text panes, menus, labels, and information messages, should be easily configurable by users. There is no way to anticipate how large is large enough. The larger fonts can be scaled, the more likely it is that users with low vision will be able to use software without additional magnification software or hardware. Although many users employ screen magnification hardware or software to enlarge their view, performance and image quality are improved if larger font sizes are available prior to magnification.

A related problem for users with low vision is their limited field of view. Because they use large fonts or magnify the screen through hardware or software, a smaller amount of information is visible at one time. Some users have tunnel vision that restricts their view to a small portion of the screen, while others require magnification at levels that pushes much

of an interface off-screen.

A limited field of view means that these users easily lose context. Events in an interface outside of their field of view may go unnoticed. These limitations in field of view imply that physical proximity of actions and consequences is especially important to users with low vision. In addition, providing redundant audio cues (or the option of audio) can notify users about new information or state changes. In the future, operating environments should allow users to quickly navigate to regions where new information is posted.

Interpreting information that depends on color (e.g, red=stop, green=go) can be difficult for people with visual impairments. A significant number of people with low vision are also unable to distinguish among some or any colors. As one legally blind user who had full vision as a child told us, his vision is like "watching black and white TV". In any case, a significant portion of any population will be "color blind". For these reasons, color should never be used as the only source of information -- color should provide information that is redundant to text, textures, symbols and other information.

Some combinations of background and text colors can result in text that is difficult to read for users with visual impairments. Again, the key is to provide both redundancy and choice. Users should also be given the ability to override default colors, so they can choose the colors that work best for them.

## **About Blindness**

There is no clear demarcation between low vision and true blindness, but for our purposes, a blind person can be considered to be anybody who does not use a visual display at all. These are users who read braille displays or listen to speech output to get information from their systems.

Screen reader software provides access to graphical user interfaces by providing navigation as well as a braille display or speech synthesized reading of controls, text, and icons. The blind user typically uses tab and arrow controls to move through menus, buttons, icons, text areas, and other parts of the graphic interface. As the input focus moves, the screen reader provides braille, speech, or non-speech audio feedback to indicate the user's position (see Mynatt 1994). For example, when focus moves to a button, the user might hear the words "button -- Search", or when focus moves to a text input region, the user might hear a typewriter sound. Some screen readers provide this kind of information only in audio form, while others provide a braille display (a series of pins that raise and lower dynamically to form a row of braille characters).

Blind users rarely use a pointing device, and as discussed above, typically depend on keyboard navigation. A problem of concern to blind users is the growing use of graphics and windowing systems (Edwards, et al, 1992). The transition to window-based systems is an emotional issue, evoking complaints from blind users who feel they are being forced to use an environment that is not well-suited to their style of interaction. As one blind user put it, "This graphics stuff gives sighted people advantages...its user friendly...all this makes it user unfriendly for us [blind people]".

Although blind users have screen reading software that can read the text contents of buttons, menus, and other control areas, screen readers cannot read the contents of an icon or image. In the future, systems should be designed that provide descriptive information for all non-text objects. Until the appropriate infrastructure for providing this information becomes available, there are some measures that may help blind users access this information. Software engineers should give meaningful names for user interface objects in their code. Meaningful names can allow some screen reading software to provide useful information to users with visual impairments. Rather than naming an eraser graphic "widget5", for example, the code should call it "eraser" or some other descriptive name, that users will understand if spoken by a screen reader.

Without such descriptive information, blind or low vision users may find it difficult or impossible to interpret unlabeled, graphically labeled, or custom interface objects. Providing descriptive information may provide the only means for access in these cases. As an added selling point to developers, meaningful widget names make for code that is easier to document and debug.

In addition to problems reading icons, blind users may have trouble reading areas of text that are not navigable via standard keyboard features. In OpenWindows and MS Windows, for example, it is not possible to move the focus to footer messages. If this capability were built into the design, then blind users could easily navigate to footer messages in any application and have their screen reading software read the content.

**TABLE 3. Assistive Technologies for Low Vision and Blind Users**

<b>Assistive Technology</b>	<b>Function Provided</b>
Screen Reader Software	Allows user to navigate through windows, menus, and controls while receiving text and limited graphic information through speech output or braille display.
Braille Display	Provides line by line braille display of on-screen text using a series of pins to form braille symbols that are constantly updated as the user navigates through the interface.
Text to Speech	Translates electronic text into speech via a speech synthesizer.
Screen Magnification	Provides magnification of a portion or all of a screen, including graphics and windows as well as text. Allows user to track position of the input focus.

### **About Hearing Disabilities**

People with hearing disabilities either cannot detect sound or may have difficulty distinguishing audio output from typical background noise. Because current user interfaces rely heavily on visual presentation, users with hearing related disabilities rarely have serious problems interacting with software. In fact, most users with hearing disabilities can use off-the-shelf computers and software. This situation may change as computers, telephones, and video become more integrated. As more systems are developed for multimedia, desktop videoconferencing, and telephone functions designers will have to give greater consideration to the needs of users with hearing impairments.

Interfaces should not depend on the assumption that users can hear an auditory notice. In addition to users who are deaf, users sitting in airplanes, in noisy offices, or in public places where sound must be turned off also need the visual notification. Additionally, some users can only hear audible cues at certain frequencies or volumes. Volume and frequency of audio feedback should be easily configurable by the user.

Sounds unaccompanied by visual notification, such as a beep indicating that a print job is complete, are of no value to users with hearing impairments or others who are not using sound. While such sounds can be valuable, designs should not assume that sounds will be heard. Sound should be redundant to other sources of information. On the other hand, for the print example above, it would be intrusive for most users to see a highly visible notification sign whenever a printout is ready. Visual notices can include changing an icon, posting a message in an information area, or providing a message window as appropriate.

Again, the key point here is to provide users with options and redundant information. Everybody using a system in a public area benefits from the option of choosing whether to see or hear notices. When appropriate, redundant visual and audio notification gives the information that is necessary to those who need it. If visual notification does not make sense as a redundant or default behavior, then it can be provided as an option.

Other issues to consider include the fact that many people who are born deaf learn American Sign Language as their first language, and English as their second language. For this reason, these users will have many of the same problems with text information as any other user for whom English is a second language, making simple and clear labeling especially important.

Currently, voice input is an option rather than an effective general means of interaction. As voice input becomes a more common method of interacting with systems, designers should remember that many deaf people have trouble speaking distinctly, and may not be able to use voice input reliably. Like the other methods of input already discussed, speech should not be the only way of interacting with a system.

**TABLE 4. Assistive Technologies for Hearing Disabilities**

<b>Assistive Technology</b>	<b>Function Provided</b>
Telecommunications Device for the Deaf (TDD)	Provides a means for users to communicate over telephone lines using text terminals.
Closed Captioning	Provides text translation of spoken material on video media. Important computer applications include distance learning, CD-ROM, video teleconferencing, and other forms of interactive video.
ShowSounds	Proposed standard would provide visual translation of sound information. Non-speech audio such as system beeps would be presented via screen flashing or similar methods. Video and still images would be described through closed captions or related technologies. This capability would be provided by the system infrastructure.

## Design Guidelines

We have taken the design issues discussed in this chapter and condensed them into a list of guidelines contained in Table 5. This table also indicates which users are most likely to benefit from designs that follow these guidelines.

**TABLE 5. Design Guidelines**

<b>Design Guideline</b>	<b>Physical</b>	<b>RSI</b>	<b>Low Vision</b>	<b>Blind</b>	<b>Hearing</b>
Provide keyboard access to all application features	X	X	X	X	
Use a logical tab order (left to right, top to bottom or as appropriate for locale)	X			X	
Follow key mapping guidelines for the local environment	X	X	X	X	
Avoid conflicts with keyboard accessibility features (see Table 6)	X			X	
Where possible, provide more than one method to perform keyboard tasks	X	X			
Where possible, provide both keyboard and mouse access to functions	X	X	X	X	
Avoid requiring long reaches on frequently performed keyboard operations for people using one hand.	X	X			
Avoid requiring repetitive use of chorded keypresses	X	X			
Avoid placing frequently used functions deep in a menu structure	X	X	X	X	
Do not hard code application colors			X		
Do not hard code graphic attributes such as line, border, and shadow thickness			X		
Do not hard code font sizes and styles			X		
Provide descriptive names for all interface components and any object using graphics instead of text (e.g, palette item or icon)				X	
Do not design interactions to depend upon the assumption that a user will hear audio information					X
Provide visual information that is redundant with audible information					X
Allow users to configure frequency and volume of audible cues			X	X	X

## Existing Keyboard Access Features

Designers of Microsoft Windows, Macintosh, and X Windows applications should be aware of existing key mappings used by access features built into the Macintosh and X Windows (and optionally available for MS Windows). These features provide basic keyboard accessibility typically used by people with physical disabilities (see table 2).

In order to avoid conflicts with current and future access products, applications should avoid using the key mappings indicated in Table 6.

**TABLE 6. Reserved Key Mappings**

<b>Keyboard Mapping</b>	<b>Reserved For</b>
5 consecutive clicks of shift key	On/ Off for StickyKeys
Shift key held down 8 seconds	On/Off for SlowKeys and RepeatKeys

**TABLE 6. Reserved Key Mappings**

<b>Keyboard Mapping</b>	<b>Reserved For</b>
6 consecutive clicks of control key	On/Off for screen reader numeric keypad
6 consecutive clicks of alt key	Future Access use

## **INFRASTRUCTURE FOR ACCESS**

We have identified a number of approaches to improve the accessibility of applications, but these approaches leave many open issues. Even if all of these strategies were adopted for every current and future interface design, accessibility is ultimately limited by the capabilities of system infrastructure to support communication between assistive technologies and applications.

Recently, some progress has been made towards providing assistive access infrastructure. Parts of the OS/2 operating system, for example, were designed with blind access in mind (Emerson, Jameson, Pike, Schwerdtfeger, and Thatcher, 1992). The operating system infrastructure allows the IBM Screen Reader for Presentation Manager to obtain information about which window is receiving input, the contents of that window, and the layout of that window. For blind users, this infrastructure support means that they can obtain screen information that might be inaccessible on other systems (including font sizes and styles as well as icon positions and labels).

In its most recent release, the X Window system has incorporated some similar access design features for users who are blind or have a physical disability (Edwards, Mynatt, and Rodriguez, 1993; Walker, Novak, Tumblin, and Vanderheiden, 1993). As a consequence, any developer of screen reading software can now develop X Window screen readers that can determine the state and content of windows.

Vanderheiden (1992b) proposed a standard cross-platform method for providing visual presentation of auditory information on computers. This strategy is based on a “ShowSounds” flag which would be set by the user. The ShowSounds capability would allow users who are deaf, hearing impaired, or in quiet environments to request all information presented auditorially to be presented visually. Multimedia applications, for example, would provide captioning for voice while alternative visual presentations would be available for non-speech audio. Such capabilities would benefit users learning a language, working in multilingual environments, as well as deaf users.

### **Future Directions**

ShowSounds is an example of the system-level approach to providing access that is the key to improving accessibility. While current direct and assistive access capabilities are positive steps, much more can be done to improve accessibility. In the remainder of this section, we discuss some of the areas where system infrastructures must be developed to provide support for access.

Users with low vision, for example, typically view a relatively small portion of their display. Often they do not see color, and because of high magnification are unable to see new information or interface changes that are occurring off-screen. In the future, users should have the capability to quickly navigate to regions where new information is posted. Future design efforts also need to explore how interfaces can gracefully handle a wider variety of viewing environments.

Note that some desktop environments (notably the Macintosh) do not provide comprehensive keyboard navigation capabilities allowing tab and arrow navigation through menus and controls. Assistive technologies requiring control navigation operate in these environments by providing their own keyboard navigation. This approach increases the potential for incompatibilities with application key functions, and may not work effectively across all applications.

MS Windows, OpenWindows, and Motif all provide keyboard navigation, but none of these environments provides a standard method for navigating to read-only interface regions such as labels, message bars, and footer messages, nor do they provide a method for obtaining information about graphics. As a near-term solution, systems need to allow users to navigate to read-only text areas, and provide descriptive information for all non-text objects. Over the longer term, systems should be designed so they present and accept information in multiple modalities.

Blind users, for example, would benefit from both speech and non-speech audio information based on video and graphics content. As in the case of closed captioning, such alternative forms of information presentation provide much more than access for people with certain disabilities. Audio descriptions of visual information would allow many users to access and manipulate graphical and video information. over conventional telephones or while working on eyes-busy tasks.

Current interfaces are primarily visual, but as telephones become integrated with computers and multimedia becomes ubiquitous, access barriers for hearing impaired users will increase. Future systems should include support of closed captioning of video. Closed captioning is useful not only for users with hearing disabilities, but also for users learning a second language and for a variety of contexts in which multiple languages are used. In addition, the descriptive text accompanying closed captioned video can be indexed to allow full-text searching of video and audio sources.

Looking farther in the future, adaptive interfaces along with intelligent agents can benefit users with disabilities by allowing them to interact with systems in the way best suited for each user (see Kuhme, 1993; Puerta, 1993). Unlike today's computers, future systems will allow most users to step up to any system and use it in a way best suited to their individual needs. We claim that an essential requirement for making this kind of adaptive system possible is to ensure that it accounts for the needs of users with disabilities.

## CONCLUSIONS

As with any other issue in human-computer interaction, the key to understanding the problem of accessibility is to understand the needs of users. We have suggested that users

with disabilities are generally not considered in interface design. A shift from the narrow view of who constitutes “the user” to a broad view is the first step towards improving the accessibility of human-computer interaction for all users. Towards this end, we have presented guidelines to increase the accessibility of applications designed for current environments. While much can be done by simply paying attention to the needs of users with disabilities, significant progress towards computer accessibility awaits improvements in the overall accessibility of the user environments.

Much is understood about the needs of users with disabilities that is not addressed by current systems. It is well understood, for example, that blind users need auditory and braille access to graphic information, that deaf users need visual access to audio information, and that users with physical disabilities often need alternative means of interacting with their systems. To address these needs, future operating system infrastructures must be designed with accessibility in mind. To the extent possible, users should be able to “step up” to systems and use them without any adjunct assistive software or hardware. Where assistive software or hardware is required, system infrastructure should provide support for it to interoperate transparently with existing applications.

Multimedia and related future technologies hold both danger and promise for future prospects of accessible human-computer interaction. The danger is that without realizing the potential roadblocks presented by these emerging audio and video technologies, interface barriers far more daunting than those today will be designed into future applications and environments. Many users could be locked out by interfaces that are usable only by people with a wide range of sensory and motor capabilities.

The promise is that these same mix of audio and video capabilities can significantly increase the accessibility of human-computer interaction. By providing the appropriate infrastructure to support multiple redundant input and presentation of information, systems of the future can allow users with disabilities to interact on their own terms while providing information in the form most useful to them. Not only will these users benefit, but all users will benefit from interacting with systems based on their needs and context.

## REFERENCES

Arons, Barry. *Tools for Building Asynchronous Servers to Support Speech and Audio Applications*, 5th Annual Symposium on User Interface Software and Technology, 71-78. ACM Press, 1992.

Blattner, M. M., Glinert, E.P., Jorge, J.A., and Ormsby, G.R., *Metawidgets: Towards a theory of multimodal interface design*. Proceedings: COMPASAC 92, pp 115-120, IEEE Press, 1992.

Brown, C. *Computer Access in Higher Education for Students with Disabilities*, 2nd Edition. George Lithograph Company, San Francisco. 1989.

Brown, C. *Assistive Technology Computers and Persons with Disabilities*, *Communications of the ACM*, pp 36-45, 35(5), 1992.

- Casali, S.P., and Williges, R.C., Data Bases of Accommodative Aids for Computer Users with Disabilities, *Human Factors*, pp 407-422, 32(4), 1990.
- Church, G., and Glennen, S. *The Handbook of Assistive Technology*, Singular Publishing Group, Inc, San Diego, 1992.
- Edwards, W. K., Mynatt, E. D., Rodriguez, T. The Mercator Project: A Nonvisual Interface to the X Window System. *The X Resource*. O'Reilly and Associates, Inc. April, 1993.
- Edwards, A., Edwards, E., and Mynatt, E. *Enabling Technology for Users with Special Needs*, InterCHI '93 Tutorial, 1993.
- Elkind, J. The Incidence of Disabilities in the United States, *Human Factors*, pp 397-405, 32(4), 1990.
- Emerson, M., Jameson, D., Pike, G., Schwerdtfeger, R., and Thatcher, J. *Screen Reader/PM*. IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 1992.
- Fels, D., Shein, G.F., Chignell, M.H., and Milner, M. Feedback Control: Whose Job is it Anyway?, pp 81-84, 25th Annual Conference of the Human Factors Association of Canada, 1992.
- Furger, Roberta. Danger at Your Fingertips. *PC World*, pg. 118, 11(5), 1993.
- Glinert, E.P., and York, B.W. Computers and People with Disabilities, *Communications of the ACM*, pp 32-35, 35(5), 1992.
- Griffith, D. Computer Access for Persons who are Blind or Visually Impaired: Human Factors Issues. *Human Factors*, pp 467-475, 32(4), 1990.
- Kuhme, T. A User-Centered Approach to Addaptive Interfaces. *Proceedings of the 1993 International Workshop on Intelligent User Interfaces*. pp 243-246. Orlando, FL. New York: ACM Press 1993.
- Lazzaro, Joseph J. *Adaptive Technologies for Learning and Work Environments*. American Library Association, Chicago and London, 1993
- Lewis, C., and Rieman, J. *Task-Centered User Interface Design*. Electronic Shareware Publication, 1993.
- Managing Information Resources for Accessibility*, U.S. General Services Administration Information Resources Management Service, Clearinghouse on Computer Accommodation, 1991.
- McCormick, John A. *Computers and the American's with Disabilities Act: A Manager's Guide*. Windcrest, 1994.
- McMillan, W.W. *Computing for Users with Special Needs and Models of Computer-Human Interaction*. Conference on Human Factors in Computing Systems, CHI '92, pp. 143-148. Addison Wesley, 1992.

Mynatt, E. *Auditory Presentation of Graphical User Interfaces* in Kramer, G. (ed), *Auditory Display: Sonification, Audification and Auditory Interfaces*, Santa Fe. Addison-Wesley: Reading MA., 1994.

Newell, A.F., and Cairns, A. Designing for Extraordinary Users. *Ergonomics in Design*, October, 1993.

Nielsen, J. *Usability Engineering*. Academic Press, Inc., San Diego. 1993.

Perritt Jr., H.H. *Americans with Disabilities Act Handbook*, 2nd Edition. John Wiley and Sons, Inc, New York, 1991.

Puerta, A.R. The Study of Models of Intelligent Interfaces. *Proceedings of the 1993 International Workshop on Intelligent User Interfaces..* pp. 71-80. Orlando, FL. New York: ACM Press

Quindlen, T.H., Technology Outfits Disabled Feds for New Opportunities on the Job, *Government Computer News*, 1993.

Sauter, S.L., Schleifer, L.M., and Knutson, S.J. Work Posture, Workstation Design, and Musculoskeletal Discomfort in a VDT Data Entry Task. *Human Factors*, pp 407-422, 33(2), 1991.

Schmandt, C. Voice Communications with Computers, *Conversational Systems*. Van Nostrand Reinhold, New York, 1993.

Tognazzini, Bruce. *Tog on Interface*. Addison-Wesley Publishing Company, Inc., 1992.

Vanderheiden, G.C., Curbcuts and Computers: Providing Access to Computers and Information Systems for Disabled Individuals. *Keynote Speech at the Indiana Governor's Conference on the Handicapped*, 1983.

Vanderheiden, G.C., Thirty-Something Million: Should They be Exceptions? *Human Factors*, 32(4), 383-396. 1990.

Vanderheiden, G.C. *Accessible Design of Consumer Products: Working Draft 1.6*. Trace Research and Development Center, Madison, Wisconsin, 1991.

Vanderheiden, G.C. *Making Software more Accessible for People with Disabilities: Release 1.2*. Trace Research and Development Center, Madison, Wisconsin, 1992a.

Vanderheiden, G.C. *A Standard Approach for Full Visual Annotation of Auditorially Presented Information for Users, Including Those Who are Deaf: ShowSounds*. Trace Research & Development Center, 1992b.

Walker, W.D., Novak, M.E., Tumblin, H.R., Vanderheiden, G.C. Making the X Window System Accessible to People with Disabilities. *Proceedings: 7th Annual X Technical Conference*. O'Reilly & Associates, 1993.

## **Sources for more information on Accessibility**

Clearinghouse on Computer Accommodation (COCA) 18th & F Streets, NW Room 1213 Washington, DC 20405 (202) 501-4906

*A central clearinghouse of information on technology and accessibility. COCA documentation covers products, government resources, user requirements, legal requirements, and much more.*

Sensory Access Foundation 385 Sherman Avenue, Suite 2 Palo Alto, CA 94306 (415) 329-0430

*Non-profit organization consults on application of technology "to increase options for visually and hearing impaired persons". Publishes newsletters on assistive technology.*

Trace Research and Development Center S-151 Waisman Center 1500 Highland Avenue Madison, WI 53528 (608) 262-6966

*A central source for the current information on assistive technologies as well as a major research and evaluation center. Trace distributes databases and papers on assistive technology and resources.*

## **Conferences**

CSUN Conference on Technology and Persons with Disabilities. Every Spring in Los Angeles, California in the USA. Phone: (818) 885-2578

Closing the Gap Conference on Microcomputer Technology in Special Education and Rehabilitation. Every Fall in Minneapolis, Minnesota in the USA. Phone: (612) 248-3294

RESNA. Conference on rehabilitation and assistive technologies. Every Summer. Location varies. Phone: (703) 524-6686

ASSETS. Annual meeting of ACM's Special Interest Group on Computers and the Physically Handicapped. Every Fall. Location varies. Phone: (212) 626-0500