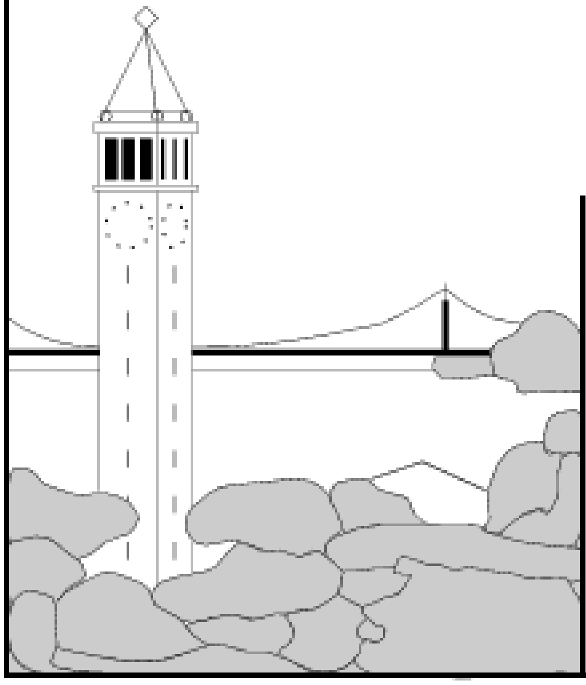
EASE: A Simulation Tool for Accessible Design

Holly Fait and Jennifer Mankoff



Report No. UCB/CSD-3-1260

August 2003

Computer Science Division (EECS)
University of California
Berkeley, California 94720

ABSTRACT

User studies involving users with disabilities often incur greater financial and complexity costs than those involving general populations. Developers of end-user systems rarely, if ever, test their applications with disabled users in the early stages of design, typically preferring instead to interview a few typical, non-disabled users, or simply conduct an informal expert review. This is problematic since the biggest design decisions are often made early in the design process. Without feedback or input from disabled users about a system, design decisions may not reflect their needs. In particular, users with vision or motor impairments may bring special requirements to the user interface design process. This paper examines the feasibility of simulating the interaction experiences of users with motor impairments to help developers identify disability-related usability problems similar to those received during user testing. We present EASE (Evaluating Accessibility through Simulation of user Experience), a discount evaluation tool that simulates the interaction experiences of users with motor impairments. We show that the use of simulation in the context of users of word prediction software, with motor impairments, is an effective approach to obtaining results similar to those found through actual user studies with motor impaired users. Further, we discuss some of the advantages simulation of this sort affords, the design and implementation of EASE, and some on-going research into other possibilities for simulation.

Categories and Subject Descriptors

H.1.2 [Models and Principles]: User Modeling and Accessibility–*evaluation/methodology, prototyping, user-centered design.*

General Terms

Design, Human Factors.

Keywords

Simulation, User Modelling, Assistive Technologies, Disability

1. INTRODUCTION

Accessibility of electronic information and interfaces for all users is of undeniable import. However, accessibility is often not extended to users with disabilities, despite the existence of legal and social pressures to do so. For instance, using most web pages is not possible, or exceedingly difficult for blind users [3]. Even simple tasks such as filling out a web form can cost a user with a motor impairment significantly more time and effort than an able-bodied user. Consider the case where a designer asks a user to input mailing, shipping and billing addresses, but doesn't offer the option to use the same data for all three. This poses a minor inconvenience for an able-bodied user, but could take a user with a sever motor impairment who has a particularly slow typing speed many times longer than necessary, simply due to poor design.

Although blind access issues have received more attention in the past, we chose focus on accessibility for users with motor impairments in order to motivate our idea of simulating user interaction experience for early stage design evaluation. Creating accessible interfaces for users with motor impairments is particularly difficult due to the various effects motor impairments have on interaction techniques, and the broad variation in how much a motor impairment affects input bandwidth. Additionally, blind interaction experience can be simulated with off-the-shelf tools (turn off the monitor, and use a screen reader), while no simulation tools exist for motor impairments. We present EASE (Evaluating Accessibility through Simulation of user Experience), a tool that simulates user interaction experiences for motor impairments. We also present a study showing that simulation, using EASE, is an effective solution for gaining feedback on user experiences.

There exists a wide range of conditions that affect motor control such that assistive devices or rate enhancement schemes are considered beneficial for users with those impairments. These disabilities include "paralysis, weakness, contractures, amputations, tremors, spasticity, and other limitations related to coordinated movement" [14]. The impairments described cause gross motor impairments, which result in "limitations in strength, coordination, and joint range of motion", fine motor impairments, which affect "precise movements of the wrist and fingers", and mobility impairments, which "may affect a person's ability to walk"[14]. Fine and gross motor impairments particularly affect a user's ability to interact with their computers.

Various software, standards and physical devices are available in order to improve accessibility for users with motor impairments as discussed above. Although they tend to focus more on visual impairments, the W3C guidelines [21] and automatic verification tools for the web, like Bobby [2] address motor impairments as well. Unfortunately, standards are typically characterized by complex or vague guidelines that can lead to multiple, conflicting interpretations. Further, it is not proven that automated tools for accessibility, nor standards alone, are adequate for insuring accessibility for disabled users [7]. Physical devices, like keyboard guards, mouth sticks and specialized switches, and applications like screen readers, word predictors and scanning or single-switch interfaces are just a few devices a person with a motor impairment might employ. However, they represent "after-the-fact" approaches and do not ensure accessibility of a product or application. This fact is easily verified by comparing how a non-impaired user is able to interact with a computer to how some individuals with disabilities are forced to interact through a smaller set of techniques. Using the example above, if an able bodied user is filling out a web form, they would usually type, whereas a user with a sever motor impairment might be limited to a scanning interface using only a single switch to enter each character. While there will always be differences in interaction techniques between disabled users and non-disabled users, the goal of accessibility can not be discarded on this basis alone.

Thus, we argue that even with the current set of accessibility guidelines, software and physical devices for users with motor impairments, user feedback about accessibility is still necessary to support accessible design. While usability testing and user studies are necessarily the backbone of user-centered design and development, user testing with special populations is often more difficult than studies with larger populations [3], [4]. Also, because of its high overhead, user testing is rarely used in the early stages of design, when interviewing a few typical, non-disabled users, or simply conducting an informal expert review is a more effective use of resources [20]. Thus, we present the idea of simulating user interaction experiences for designers.

We present a tool for simulating the interaction experiences of users with motor impairments for designers to evaluate and improve the accessibility of their products early in the design cycle. Simulating user interaction experiences (referred to as "simulation" from now on) provides numerous benefits to designers and to communities seeking accessible products. A flexible and robust tool affords early and frequent evaluation of applications, taking into account user experience, *i.e.*, it allows developers to use their products in ways that disabled users might. For instance, a designer who attempts to interact with her application using a single-switch input and scanning interface, will better understand the effort a user with a motor impairment must exert to use the application. We believe that using a simulation tool to identify accessibility issues early in the design process can improve the quality and payoff of user studies later in the design process, and the overall accessibility of the final design.

In order to deem simulation a viable technique, a developer using it must be able to find results similar to those he might find with disabled users. To test the feasibility of simulation as an evaluation tool, we compare the results between three studies of word prediction (WP) software. We show that the use of simulation in the context of users of WP software is an effective approach to obtaining results similar to those found through actual user studies with motor-impaired users.

We chose to examine WP software with simulation because of the plethora of information regarding user text entry rates, keystroke savings, and error rates, as well as the utility it can offer users with motor impairments. This field is particularly appropriate to consider since early testing, had it included actual users with motor impairments, would have uncovered the small or negative benefit most motor-impaired users experience with WP software. Research that does not involve users tends to focus on keystroke savings (based on the prediction scheme or presentation used), or upper bounds on prediction rates and prediction accuracy given a WP configuration [1], [5], [13].

The next section presents an overview of related work, particularly pertaining to the question of how and on what basis motor impairments should be simulated. This is followed by a description of EASE, our simulation tool, in Section 3. Section 4 presents two studies demonstrating that simulation offers accurate results. Section 5 concludes the paper, and discusses our plans for the future involving a consideration of other domains where simulating user experience could improve designing for accessibility.

2. BACKGROUND

Usability and accessibility of interfaces is clearly paramount for disabled users. One clear solution to understanding how to make an application accessible to users with disabilities is usability testing. Through user studies, Koester was able to examine users of WP technologies, and found that most users will not experience an increase in their text entry rate, even with specialized customizations, unless their current text entry rate is less than 8 words per minute (WPM) [11]. Readers should note that prior to studies like this with feedback and data from disabled users, keystroke savings were (and still are) touted as the gold standard for WP technologies. Some of these applications report keystroke savings of 37-47% [10]. Note that keystroke savings do not imply increased text creation, or an increase in time savings for the user. Reports of increased keystroke savings do not imply that users will experience any benefit beyond typing fewer characters. Even the developers of the WP system used for our studies claim keystroke savings of up to 46% for unfamiliar words, and 77% for familiar words [1]. One of the main differences between these findings and the findings from user studies is that keystroke savings measurements do not take into account the additional cognitive load required for users of systems that, like word prediction, reduce motor requirements [12]. Even when keystroke savings are possible, these savings are often outweighed by the additional costs of scanning a word prediction list, choosing the correct word from the list, or not finding the word in the list and then typing another character before reading the list again.

This small example shows that user testing is a necessary component of the design process. Unfortunately, user testing with special populations often requires greater effort, time and monetary commitments on the part of designers, developers and their companies than user testing with non-disabled populations [3], [4]. Further, while user testing is ideal for user-centered design, it is not the standard. Vredenburg *et al.* found that most developers tend to prefer expert and informal evaluation for early design phase evaluation [20]. Actual user studies, if they happen at all, only happen towards the end of the design phase. Similarly, Gould and Lewis found that even if designers felt they understood good design practices, they still more often relied on their intuitions for design, than the principles they thought they understood [6].

This presents a problem. User testing can improve design, but is often too expensive, time consuming, or difficult to perform as often as is needed throughout the design process. One alternative is to employ user modeling. Keates *et al.* examined the feasibility of user modeling for motor-impaired users. They concluded that since most motor impairments are too broadly defined and have too many associated effects on motor abilities, no extensible "implicit or explicit models of user behavior" are available to create an effective user model [8]. Thus, a complete user model for simulation is not necessarily the best approach. Trewin developed an interaction model for motor impaired users [19]. She used her data to develop a number of generalizations about the types of errors motor impaired users experience during text entry, on which we base the design of our simulation tool. This work points clearly to the possibility of simulating generalized input patterns for the purpose of accessibility testing in early stage design phases.

Koester and Levine also showed that modeling works on a smaller scale [9]. They examined the feasibility of simulating user text entry times with a Keystroke Level Model (KLM). With this model, they found that "user performance with word prediction systems can be successfully modeled using a relatively simple model that considers only keypress and list search actions" [10]. Further, they developed a user performance model to determine whether the cost of a WP tool outweighs its benefit [9]. These findings show that GOMS/KLM is an appropriate approach for discovering possible usability issues without incorporating user testing in early design stages. However, a new GOMS/KLM model must be built for each interface, and the expertise required to build a GOMS/KLM model is often greater than the effort a developer is willing to put into user testing. The difficulties associated with GOMS as an evaluation technique thus leave it outside of the abilities of many developers. From the perspective of using simulation as a discount evaluation tool, Koester and Levine's findings are quite encouraging. Koester and Levine were able to accurately simulate and predict user input times in a narrowed context; therefore, it follows that offering a more flexible tool for simulation than strict modeling is a potentially effective approach to understanding the usability of a product or application. We will revisit Koester and Levine's discussions, study and findings from [10] in Section 4.3 to compare with our user study results.

Given these results, it is apparent that full-fledged user modeling, like user testing, is not an appropriate solution for a discount evaluation tool. Simulation presents itself as a middle ground between these two successful, but often neglected, solutions.

3. EASE

In this section we discuss the design and architecture of EASE, and some of the benefits afforded by simulation of user interaction experience. EASE stands for "Evaluating Accessibility through Simulation of user Experience."

3.1 EASE Architecture

EASE includes functionality for modeling two common effects of motor impairments on keyboard input: *reduced input bandwidth* and *increased error* (or adding errors to a user's typical input stream).

3.2 Data Used to Define Simulations

The *reduced input bandwidth* mode in EASE takes a purely user-configurable approach due to the wide range of input rates associated with different motor impairments. Further, data on the average input rates for users with motor impairments varies so greatly, that a user-configurable approach is the best way to insure robustness and usability of EASE.

The *increased error mode* in EASE is derived from Trewin's study of the most common types of keyboard and mouse errors for individuals with varying motor impairments [18]. In Trewin's work, participants with motor impairments where chosen such that the effects of their impairments covered a broad range of mouse and keyboard input difficulties. The most common errors she found include *Additional Key* and *Bounce* errors. *Additional Key* errors occur when a user hits a key adjacent to the key they intended to press. *Bounce* errors occur when a user presses a key for longer than the timeout allowed between the key_down and key_up events.

The percentage of error to include depends greatly on the user being modeled. Trewin found error rates ranging from less than 1% to 9% for users with differing motor impairments. Research on input errors for disabled users is not widely available, nor is it widely generalizable due to the atypical nature of disability [8], [18]. Trewin cites the tendency of HCI research to focus on "cognitive errors and their causes, ignoring physical errors of the kind produced by erratic motor control" as one predominant reason why data of this type is unavailable. However, the generalizations Trewin found provided an excellent basis for the design of *Additional Key* and *Bounce* errors in EASE.

3.2.1 Reduced Input Bandwidth

Reduced input bandwidth is achieved by forcing a user to wait a constant number of seconds between each keystroke. EASE currently supports changing input bandwidth on the fly between four preset input rates. Users of EASE simply type the access code for the rate desired to turn the rate throttle on and off. The preset input rates are 5, 8, 12 and 15 words per minute (WPM), but are further configurable *via* the EASE source code.

3.2.2 Bounce Errors

Bounce errors are quite straightforward. When a user enters a key and the doubling error mode is on, that key is repeated in the output 10% of the time. The percentage of the time doubling errors occur is available in the code and is configurable depending on the user experience being simulated and the effect of the motor impairment expected.

3.2.3 Additional Key Errors

For additional key errors, we use an error map, or a confusion list [15], to determine the key to output. The error map is an adjacency list with the error keys and error probabilities for each entry in the array. This map consists of an array of key scan codes, where each key points to an array of scan code and probability pairs. Each array is the set of keys adjacent to the indexing key, paired with the set of probabilities for each key (see Figure 1.) Given the key pressed and a random integer between 0 and 99, the key to output is chosen by indexing into the letter array and then summing all of the probabilities until the sum is greater than the random number. The letter being pointed at that moment is the letter that replaces the index letter. For example, if a user presses the 'a' key, then an 'a' is drawn to the screen 64% of the time, 's' and 'q' are drawn to the screen 6% of the time, and so on. Similarly, if a user presses the 'a' key, and the error index is 74, then a 'q' is drawn to the screen instead of an 'a'. The current implementation only includes the keys directly adjacent a given key in its error list.

3.3 Implementation of EASE

EASE is implemented as a device driver based on Ctrl2Cap, a kernel-mode WDM[1] device driver developed by System Internals [17]. Ctrl2Cap "layers in the keyboard class device's stack above the keyboard class device" in order to catch and filter keyboard read requests [17]. For each request, it posts an I/O completion callback, at which point it processes the scan code being returned. Each of the error modes is easily turned on or off by using the associated "start" and "stop" codes. For instance, the start code string for decreased input bandwidth is "ZZZ", and the stop code is "QQQ". When an error mode is turned on, then the read request is changed accordingly. The codes to control the on/off states of each mode are quite easy to change in the driver, however, future work will allow users to customize the start and stop codes.

3.4 Benefits of EASE

The flexibility EASE affords changing and configuring different parameters to simulate a user interaction experience is necessary for such a tool to work effectively. Currently, all of the error

[1] The "Windows Driver Model (WDM) is a strategy for making driver development simpler" [16].

states are configurable (on/off), and can be used in conjunction with one another; further, the reduced input bandwidth mode is configurable with a variety of different input rates. With this flexibility, designers can test their systems with a variety of input conditions, including reduced input bandwidth and different types of increased error. This functionality also allows designers to understand a greater range of user experiences and create specific simulations and more robust designs.

There are further benefits developers can gain from using a simulation tool like EASE. EASE allows designers to experience what their users might experience every day while trying to interact with their computers. An understanding of the inaccessibility struggles users face gives designers an experiential knowledge from which to design in the future. Furthermore, designers do not design in a vacuum. As one designer begins to understand the accessibility problems one group of users might face, they can share these discoveries with other designers and developers for incorporation into their designs.

Another benefit of using a tool like EASE for early interface evaluation involves user studies. Since EASE is not intended as a replacement for user studies, it is assumed that after early and iterative design testing, at some point, designers will ask disabled users what they think of the system. By this time, developers will have already found and fixed a number of usability and accessibility problems in their systems. It follows that their user studies can then focus on problems that would not have been discussed if the original problems had not been fixed. This benefits both designers and users. Designers will not leave user studies feeling that their systems are completely inaccessible due to initial, large usability problems that were not addressed early on. Users are able to focus more on usability once basic accessibility issues are out of the way. If accessibility is paramount to the developers and designers, then less time and effort is spent redesigning the system to meet accessibility standards. Further, using a tool like EASE offsets the chances of a disastrous user study, that can lead to abandonment of design or, worse yet, the goal of accessibility, due to poor initial feedback.

4. COMPARISON OF SIMULATION AND DIRECT USER FEEDBACK

We will discuss two approaches to simulation, and compare the results of user testing with these simulation methods to results found in user studies with motor-impaired users. Our goal is to test the effectiveness of simulation in general, and EASE in particular for measuring the effect of a motor impairment on the use of a word prediction tool. As mentioned before, we chose to focus on WP because success is so hard to predict without involving users. We compared reduced bandwidth simulated with EASE, a physical means of reducing bandwidth, and results disabled users of WP (from a study conducted by Koester and Levine [10].) Our goal was to better understand the impact of reduced input bandwidth on the usability of a commercial WP program. Our hypothesis was that physical simulation, and our simulation tool, would give results similar to Koester and Levine's study with disabled users.

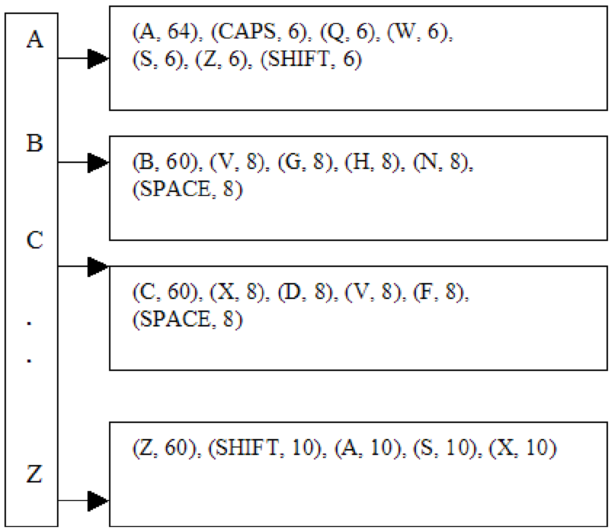

A
B
C
.
.
Z
(A, 64), (CAPS, 6), (Q, 6), (W, 6), (S, 6), (Z, 6), (SHIFT, 6)
(B, 60), (V, 8), (G, 8), (H, 8), (N, 8), (SPACE, 8)
(C, 60), (X, 8), (D, 8), (V, 8), (F, 8), (SPACE, 8)
(Z, 60), (SHIFT, 10), (A, 10), (S, 10), (X, 10)

Figure 1: Error map used to determine the key to output after an input key is pressed. For the 'A' key, there is a 64% chance that 'A' will be output, a 6% chance the CAPS lock will be activated, a 6% chance that the 'Q' key is output, and so on. Assume the 'A' key is pressed and the random number 72 is generated. Then the key output is 'Q'

We conducted two studies to measure the user experience and interaction rates of able-bodied users when using WP and either physical or software-based simulation of reduced input bandwidth, described in Section 4.1 and 4.2, respectively. We took into account qualitative and quantitative measurements of user experience and input speeds. Based on our results, we conclude in Section 4.3 that simulation is a viable approach, and that EASE is an effective tool for simulation.

4.1 Study of Physical Simulation

The first study we conducted looked at the experiences of non-impaired users with WP software where their input rates were reduced by using an unwieldy pointer when typing. This study gave us initial feedback on the feasibility of simulation.

4.1.1 Method

Participants were 6 graduate students with no known cognitive or motor impairments, each of whom demonstrated typing proficiency and was comfortable using a computer for word processing tasks. Users' input was gathered by asking them to respond to a simple essay question for each of three input conditions. In the *reduced bandwidth* condition users were asked to type using a large pointer. This pointer was a pointing stick for use with a regular keyboard and was created by the authors after initial tests showed that asking users to type with only one finger did not reduce their input bandwidth enough to offer them any benefit from the WP software. This condition measured a baseline input rate for users with narrowed input bandwidth. In the second typing condition, *word prediction*, users were asked to use WP software along with the pointer. The final typing condition was natural typing, or typing in the way most natural to the user, as an overall baseline of input rate and ability.

The study was conducted using Microsoft Word for word processing, and Aurora, a WP tool from Aurora Systems, Inc. [1]. Users were given a tutorial on how to use the WP software and were encouraged to practice with it until they felt comfortable. Users were not assigned a WP selection method, *i.e.*, users were *not told* to "always read the WP list before typing another letter or choosing a word", or "only read the WP list for words long than three letters." We used a randomized, within subjects design to account for learning effects.

4.1.2 Results

Results from this experiment show that introduction of the WP software actually reduced the input rate for most users, even in the reduced bandwidth condition. On average, users' input was 9.8 WPM (std. Dev. is 1.22 WPM) using during the *reduced bandwidth* condition, and fell to 7.2 WPM (std. dev. is 2.08 WPM) during the *word prediction* condition. Qualitatively participants in the reduced bandwidth condition were observed, and reported, typing an entire word with just the pointer, even when the appropriate word was available on the WP screen. One participant reported typing entire words only for shorter words (and words not in the WP list), and all but one of the participants were observed doing this as well. When asked how often he or she typed an entire word without using the WP list, the average participant response was 3.5, or slightly more than "sometimes" on a scale from 0 (never) to 5 (often). This result is particularly interesting since it indicates that the average input speed of these users is possibly still too great for them to consider WP software beneficial to their input rate. Quantitative results were less clear. On average, a t-test showed that users were faster with the pointer alone than with word prediction (p=.04). Visual inspection shows that not one participant exceeded their maximum input rate when using WP, although almost half of participants came close to achieving their maximum input rate.

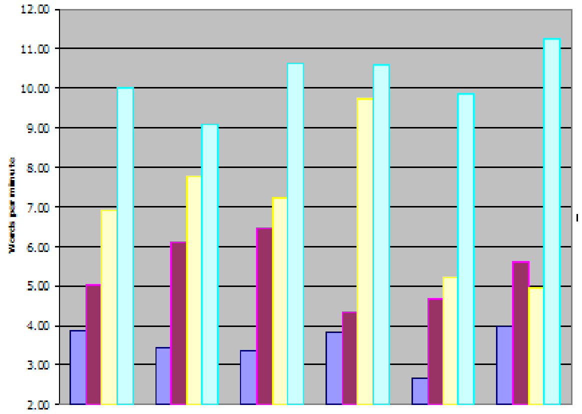


Figure 2: Input speeds achieved in each condition. Note that in each case, the line never exceeds the maximum possible speed for that condition. For instance, the 5 WPM condition for each user never exceeds the 5.00 WPM line on the Y axis.

4.2 Study of Software-Based Simulation

The second study employed EASE, described in Section 3. In this study, we used only the input rate reduction portions of the tool to put an upper limit on the maximum rate a user could type. Our goal in this study was to show that EASE was effective for simulation, and in fact a more effective tool than the physical methods employed in the first study.

4.2.1 Method

With this study, similar to the previous study, we preformed a within subjects study. Participants were 6 graduate students with no known cognitive or motor impairments. Each could demonstrate typing proficiency, and was comfortable using a computer for word processing tasks. Each participant was asked to respond to four simple essay questions, each with a different maximum input rate, and employing WP software for each response. The rates chosen were 5, 8, 12 and 15 WPM[2], in order to explore, through simulation, the threshold at which users experience benefit from WP tools. The ordering of these input speeds was randomized in order to account for learning effects. These rates capped the maximum possible speed for any keyboard input. For instance, if a user was responding to a question with a maximum rate of 8 WPM, their rate may fall well below that of 8 WPM, but could never go beyond 8 WPM unless they received benefit from the WP software. Again, the software used was Aurora, and users were given a tutorial on how to use the WP software and were encouraged to practice with it until they felt comfortable. Users were not assigned a WP selection method. We used a randomized, within subjects design to account for learning effects.

4.2.2 Results

All users, despite a WPM limit, typed more slowly than that limit allowed when using WP. Figure 2 illustrates this graphically, showing the speed achieved by each user in each condition.

At higher input speeds we found an interesting result. The greater the input speed, the greater the distance between a user's actual speed and the maximum possible speed. For instance, based on the 12 WPM condition, we determined with probability 99% that users, on average, will only achieve speeds of 9 WPM or less, when their input rates are 12 WPM. Similarly, based on the 15 WPM condition, we determined with probability 99% that users, on average, will only achieve speeds of 10 WPM or less, when

[2] Words per minute are calculated from characters per minute, assuming an average word length of 5 characters.

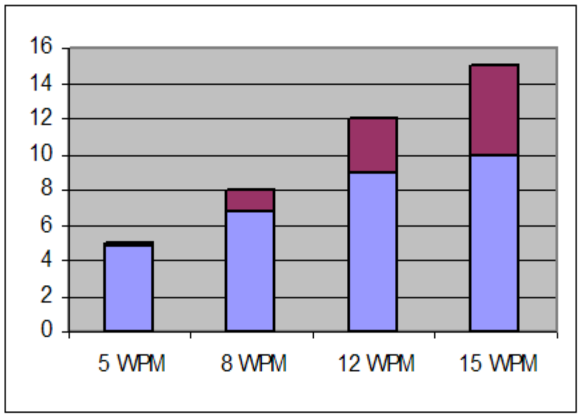


Figure 3: Percentage of maximum speed, achievable on average, compared to maximum speed, calculated using a Student's t test. The bars represent, from left to right, 5 WPM, 8 WPM, 12 WPM, and 15WPM maximum typing speeds (also shown as the top of the purple section). With 99% probability, users will only type as fast as the bottom portion of the bar when WP is available, despite the possibility of a higher maximum speed. For example, in the 8 WPM condition (2nd bar), users will achieve, on average, only 6.8, or 85% (bottom section) of maximum possible speed (top section.)

their input rates are 15 WPM. These calculations represent users achieving only 75% and 66% of maximal speed, respectively. We had similar findings for the 5 WPM and 8 WPM conditions. With probability 99%, at 5 WPM, users will on average achieve only 4.8 WPM, or 96% of maximum speed, and at 8 WPM, users will, on average, only achieve 6.8 WPM, or 85% of maximum speed. See Figure 3.

Four of our six users reported that not using word prediction, or ignoring the WP screen part of the time allowed them to input more quickly. Users also reported and were observed using the WP less effectively and only for longer words when typing at higher speeds.

4.3 Discussion

The results from our two studies display many similarities, when compared to Koester and Levine's. First, each of our studies concluded that participants with input rates greater than 8 WPM found no benefit from word prediction, and in fact were slowed down by it, even though its use was optional. Our findings are in line with Koester's anecdotal report, after working with users with motor impairments, that only users with an input rate less than 8 WPM would receive benefit from WP [11]. In our study, the results for users typing at 5 WPM were inconclusive, because input rates with and without WP were very close, and in fact we found in a small longitudinal study that users could achieve speeds greater than 5 WPM with practice when using WP.

Our results also match those of Koester and Levine's study [10], where disabled participants with reduced input bandwidth caused by motor-impairment were found to gain no benefit from using word prediction. With respect to simulation, these findings are similar to what others have found during tests with motor-impaired users: "[T]he additional cognitive and perceptual activities [required with this technology] reduce the benefit of decreased motor requirements"[10]. Further research also agrees with our findings for simulation: "For most combinations of keystroke savings…WP is likely to… enhance rate for users who type *slower* than 8 WPM" [3] [11].

An alternative to study #2 would have been to slow the user with additional key and bounce errors, in addition to limiting bandwidth. We chose not to do this so that we would not have to measure the resulting bandwidth achievable by each user in addition to measuring the bandwidth achieved when using WP. We also chose not to enforce a selection rule for the WP software (i.e., "always read the list before making a selection or typing" or "only read the list for words longer than three letters", etc.) This gives results more similar to those one might find in real user situations.

In terms of what is known about the cognitive load associated with word prediction, we found signs that users in our simulation conditions experienced the same cognitive load issues as are typically observed for other users of word prediction software. While not surprising, it was important to validate this. Also typical of users of word prediction, users in our simulation conditions were observed typing an entire word, despite its availability on the WP list.

EASE provided several benefits over physical simulation. First, it allowed us to control user input rates at a much finer granularity (users in our first study varied from seven to 16 WPM using the same simulation tool).

In summary, the similarities between our software-based simulation and Koester and Levine's work are encouraging. Specifically, they support the feasibility of using simulation to find usability and accessibility problems similar to those one might find with disabled users. It follows that, given a tool like EASE, which is relatively simple to configure and use, a developer can gain information about user experience, as well as generalizable knowledge concerning typical experiences for users with motor disabilities. This knowledge is then applicable to future interface design.

One unfortunate implication of these findings is that developers might find themselves tempted to use simulation as a replacement for user testing, or as an excuse to rely on their intuitions. This approach is entirely contrary to the purpose of EASE. EASE is designed to give developers a broader understanding of what it means to rely on accessibility and to encourage the development of accessible technologies. This tool is intended to improve accessibility in early design stages and not as a replacement for user input.

5. CONCLUSION AND FUTURE WORK

The comparisons between our studies and the work of Koester and Levine, as well as others give strong evidence that simulation of user interaction experience is a feasible method to obtain results similar to those found with target users. Beyond these findings, additional benefits afforded by a tool like EASE, as discussed in section 4.2, show that simulation of user experience, while feasible, also has the opportunity to greatly improve

[3] Emphasis added here only.

knowledge of accessibility problems, and accessibility of applications.

Future work requires following four paths. The first path includes evaluation of the methodology EASE implies. Our current work examines the applicability of simulation as a tool for improving web design for blind users. A similar approach is applicable for EASE. We plan to investigate whether designers create better (more accessible) applications when they use EASE to interact with their systems in the same way a user with a motor impairment might.

The second path addresses the ease of use of EASE. A user interface (UI) is in the works to allow users to adjust different aspects of the system on the fly. With this UI, developers will control the percentage of keystrokes that they want output as Wrong Key or Doubling Errors, the error distribution and keys in the error map and the speed of the narrowed input bandwidth desired.

The third path addresses improvements in user modeling, possibly through temporal introduction of error, or basing error on common English keystroke mistakes or language constructs. This improvement would fill out the suite of simulations with mouse input simulation. Planned interaction simulations for mouse input include increased directional and amplitude errors for pointing and unplanned or unrecognized "clicks" for selection.

Finally, our findings warrant future investigation into other possible domains for simulation of user interaction experiences like web usage for blind users, simulating mouse error, or other input errors. We are also currently investigating the feasibility of this sort of simulation for developers of web pages for blind users. We hope to show that the empathic modeling approach taken with EASE is also an appropriate approach for simulating the interaction experiences of blind users, and that this simulation can lead to more accessible web pages for the blind. See section 4.4 for further discussion on these projects.

A tool like EASE has the ability to improve accessibility on a wide scale due to the ease of deployment (*i.e.*, designers do not have to leave their desks) and flexibility EASE affords. Providing an easy to use and powerful tool for designers is just another step in allowing them to create accessible and more usable applications.

6. ACKNOWLEDGMENTS

Many thanks to our willing participants for their time and feedback. Thanks also to the National Science Foundation for funding this work. Thanks to all in the IO group for their time and effort in proofing this paper.

7. REFERENCES

[1] Aurora Systems Inc. http://www.aurora-systems.com/pf/compare.html

[2] Bobby Worldwide http://bobby.watchfire.com/bobby/html/en/index.jsp

[3] Coyne, K. P. and Nielsen, J. "Beyond ALT Text: Making the Web Easy to Use for Users with Disabilities." Nielsen, Norman Group, October, 2001.

[4] Coyne, K. P. and Nielsen, J. "How to Conduct Usability Evaluations for Accessibility: Methodology Guidelines for Testing Websites and Intranets With Users Who Use Assistive Technology." Nielsen Norman Group, October, 2001.

[5] Garay-Vitoria, N. and González-Abascal, J., "Intelligent Word-Prediction to Enhance Text Input Rate (A Syntactic Analysis-Based Word-Prediction Aid for People with Severe Motor and Speech Disability)." In *Intelligent User Interfaces 1997*, pp.241-244, 1997.

[6] Gould, J.D. and Lewis, C.H. "Designing for Usability: Key Principles, and What Designers Think". *Communications of the ACM*, **28**(3): 300-311, 1985

[7] Ivory, M.Y., Mankoff, J. and Le, A. "Using Automated Tools to Improve Web Site Usage by Users with Diverse Abilities." In *IT&Society*, Special Issue on Web Navigation Skills, **1**(3):195–236, 2003.

[8] Keates, S., Clarkson, J., and Robinson, P. "Investigating the Applicability of User Models for Motion-Impaired Users." In Proceedings of *ASSETS '00*, pp. 129-136, 2000.

[9] Koester, H.H. and Levine, S.P., "A Model of Performance Cost Versus Benefit for Augmentative Communication Systems," in *Proc. 15th IEEE-EMBS Conf.*, pp.1303-1304, 1993.

[10] Koester, H.H. and Levine, S.P. "Model Simulations of User Performance with Word Prediction." In *Augmentative and Alternative Communication*, **14**(01): 25-35, 1998.

[11] Koester, H. H., "Word Prediction—When does it enhance text entry rate?" Presented at RESNA 2002, http://www.aac-rerc.com/downloads/Word_Prediction.pdf

[12] Levine, S. P., Horstmann, H. M., and Kirsch, N. L. "Performance considerations for people with cognitive impairment in accessing assistive technologies," *J. Head Trauma Rehab*, **7**(3): 46-58, 1992.

[13] Lesher, G.W., and Rinkus, G.J. "Domain-Specific Word Prediction for Augmentative Communication." In *Proceedings of the RESNA 2002 Annual Conference*. 2002.

[14] Mann, W. C. and Lane, J. P. "Assistive Technology for Persons With Disabilities." The American Occupational Therapy Association, Inc. Bethesda, MD, 1995.

[15] Marx, M. and Schmandt, C. "Putting People First: Specifying Proper Names in Speech Interfaces." In *Proceedings of ACM UIST '94*, pp. 30-37, 1994.

[16] Microsoft Windows Driver Model http://www.microsoft.com/hwdev/driver/wdm/default.asp

[17] System Internals http://systeminternals.com/ntw2k/source/ctrl2cap.shtml

[18] Trewin, S. "A Study of Input Device Manipulation Difficulties." In Proceedings of *ASSETS '96*, pp. 15-22, 1996.

[19] Trewin, S. and Pain, H. "Dynamic Modeling of Keyboard Skills: Supporting Users with Motor Disabilities." In *Proceedings of the Sixth International Conference on User Modeling*. pp. 135-146, 1997.

[20] Vredenburg, K., Mao, J., Smith, P. and Carey, T. "A Survey of User-Centered Design Practice." In *Proceedings of the CHI 2002 conference on Human factors in computing systems*, pp. 471-478, 2002.

[21] W3C Web Accessibility Initiative, http://www.w3.org/WAI/