Using IDs to Improve

Web Navigation with a Keyboard

Hooman Baradaran

A thesis submitted to the Faculty of Graduate Studies

in partial fulfilment of the requirements for the degree of

Masters of Science

Graduate Programme in Computer Science

York University

Toronto, Ontario

September 2009

USING IDS TO IMPROVE

WEB NAVIGATION WITH A KEYBOARD

by **Hooman Baradaran**

A thesis submitted to the Faculty of Graduate Studies of York University in partial fulfilment of the requirements for the degree of

MASTERS OF SCIENCE

Using IDs to Improve Web Navigation with a Keyboard

by **Hooman Baradaran**

By virtue of submitting this document electronically, the author certifies that this is a true electronic equivalent of the copy of the thesis approved by York University for the award of the degree. No alteration of the content has occurred and if there are any minor variations in formatting, they are as a result of the conversion to Adobe Acrobat format (or similar software application).

Examination Committee Members:

1. Scott MacKenzie
2. Nick Cercone
3. Wolfgang Stuerzlinger
4. Anne Moore

ABSTRACT

Two variations of a keyboard-only web navigation technique were analysed and evaluated. "ID Navigation" assigns a unique, incremental identifier (ID) to each interactive web page element. The ID is displayed as a tiny label beside each element when an activation key is pressed. The time to select elements using a mouse, TAB-key navigation, and the new technique (using numeric IDs) was analysed using a keystroke-level model and measured in an initial experiment. Selection time increased linearly with the number of elements for TAB-key navigation and was constant for ID navigation and the mouse. ID navigation was significantly faster than TAB-key navigation however the mouse was fastest overall, as expected.

In a 2-hour paid experiment two variations of the new technique were compared in more detailed. One variation was an improved version of the existing technique using numeric IDs, the other used a set of IDs limited to four sequential keys each associated with a fixed finger. The improved version was significantly more accurate while the relative differences in selection time were similar to the initial experiment. Numeric ID navigation was the faster of the new methods and was better liked by the users, especially in a task involving data entry.

هرکه ز آموختن ندارد ننگ در ز دریا برآرد و لعل از سنگ

(حکیم نظامی گنجوی)

"He who shames not at learning,

can draw forth pearls from the water, rubies from the rock."

(*Nezami Ganjavi*, Persian poet, 1141 to 1209)

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Scott MacKenzie, for his knowledge, guidance, and support and I would like to thanks my advisor, Dr. Nick Cercone, whose generous funding of my Masters program made my research possible.

Thanks also go to Dr. Wolfgang Stuerzlinger, and Dr. Anne Moore for taking time out of their busy schedules to serve on my thesis examination committee.

I would like to thank all my friends (specially Behrooz, Damon, Mona, Nariman, Nassim), colleagues and other participants for their time, effort, and helpful suggestions.

Most importantly, I would like to thank my parents and my sister Sanaz, for their love, encouragement, and support.

Table of Contents

List of Tables

List of Figures

Chapter 1
Introduction

Web browsers are a classic example of "direct manipulation interfaces". Shneiderman [28] coined this term in the 1980s to characterize the emerging style in computing, as character-based key-entry gave way to graphical point-and-click interfaces. The "directness" in these systems comes by way of the mouse, where users manoeuvre an on-screen tracker (via the mouse) to an element of their choice and then select the element by clicking a mouse button. Simple as this is, there are a number of deficiencies. One is the constant need to "home" the hand between the keyboard and the mouse. Another is the valuable desk space ("real estate") the mouse occupies beside the keyboard. Notably as well, some users have a preference (or need), and some environments mandate, a keyboard for interaction. And, so, keyboard-only interaction is of interest even today.

Web interfaces were designed as point-and-click graphical user interface systems; and, so, HTML elements [24] are naturally manipulated by pointing devices. This works well when simply browsing through web pages on a desktop computer with a mouse. However, point-and-click interaction does not work well in some situations, including when only a keyboard is used.

1.1 Keyboard Access for Disabled Users

Individuals with neuromuscular impairments often cannot use fine movements with a mouse (or use a mouse at all) due to spinal cord injuries, brain damage, and such. In fact, disability is not the only reason for decreased mouse performance. In a study of performance limitation in tracking with a mouse [25], accuracy of the mouse decreased with old age in addition to motor disabilities. This is why web accessibility guidelines [1,8] require the possibility of web browsing using a keyboard in lieu of a pointing device in their efforts to support device-independence and "universal access" [27] to the web.

Web accessibility guidelines have tried to improve keyboard navigation by improving existing keyboard navigation methods using the TAB key (see section 2.1). Unfortunately, the majority of web sites do not follow these usability standards and this trend is not improving over time [11]. Thus, an improvement to existing keyboard navigation techniques is needed that can handle web sites regardless of their structure and compatibility with usability standards.

Furthermore, any navigation technique that is based on keyboard characters, can be extended to alternative input methods such as speech recognition that make web navigation accessible to more severely disabled users who cannot even use a keyboard. For over a decade there have been efforts on using natural speech recognition for web navigation [12], where, for example, a user could read the text of a hyperlink to select it. However, despite the advances in automatic speech recognition (ASR) [13], these

systems cannot accurately adapt to a wide variety of speakers, speaking conditions and noise. Other efforts in navigation using speech, such as the Vocal Joystick [5], use voice parameters (instead of natural language) to control objects. This works more accurately in a wider variety of conditions. Similarity, a keyboard navigation technique that uses a small set of simple character-based commands can potentially be extended to support existing speech recognition systems.

1.2 Keyboard Access for Expert Users

On a typical QWERTY keyboard there are 18 "power keys" (e.g., ENTER, BACKSPACE, etc.) on the right side of the keyboard while only a handful of these keys exists on the left side. Most users hold the mouse with their right hand as well. Arguably, on a typical desktop system with a keyboard and mouse the right hand is overloaded [19]. An improved keyboard navigation method which makes use of the left hand can reduce the load on the right hand and potentially benefit experienced users.

Additionally, experienced users and touch-typists may simply prefer to use the keyboard to reduce the time lost due to "homing". Today, a growing number of applications use web-based interfaces and when using web interfaces for data entry, continuous switching between the keyboard and mouse is required. Even if the time difference to complete the task with the keyboard alone is more than the homing time needed to move the hand to the mouse and back, some users might simply find it more comfortable to access the keyboard without having to switch devices.

The efficiency of keyboard navigation is particularly important for these expert users. While keyboard navigation is possible in current web browsers, it is generally slow and inefficient [26].

1.3 Improving Keyboard Navigation Using IDs

This thesis presents a method of keyboard-only web navigation where unique, incremental identifiers (IDs) are assigned to interactive elements on the page. The IDs appear beside each interactive element when a modifier key ("activation key") is pushed. This is followed by selection with a couple of keystrokes required to select the ID. Similar methods are implemented as *Firefox* extensions (e.g., *Mouseless Browsing* – explained in section 2.4), however this method has never been studied scientifically and no empirical evaluations exist.

This interaction technique is defined in detail in Chapter 3. It is then compared to traditional TAB-key and mouse navigation using a predictive model. In Chapter 4, an initial experiment (henceforth "Experiment 1") is performed on a modified version of the available extension and the results are compared to the predictions by the predictive model. In Chapter 5, this technique is improved and reimplemented based on the results of the initial experiment. Later, a predictive model is given for a data entry task that is empirically evaluated in the following chapter. In Chapter 6, a more extensive experiment ("Experiment 2") is performed by paid participants in multiple sessions. This experiment makes use of the improved implementation which includes two variations of the

technique and includes a data entry task. To the author's knowledge, these are the first empirical tests of such a technique. In both experiments, the interaction technique is compared to traditional TAB-key and mouse navigation and the results of the empirical studies are compared to the predictive model.

Chapter 2
Background and Related Work

Navigating web interfaces using the keyboard in most browsers requires a huge number of TABs, also known as TAB-chains. To reach a desired element, users press TAB to advance through all other selectable elements between the element currently focused and the target element. In a typical web page, there are two types of "selectable elements": widgets (buttons, text-input elements, drop-down lists, check-boxes and radio buttons, etc.) and hyperlinks. See Figure 1.

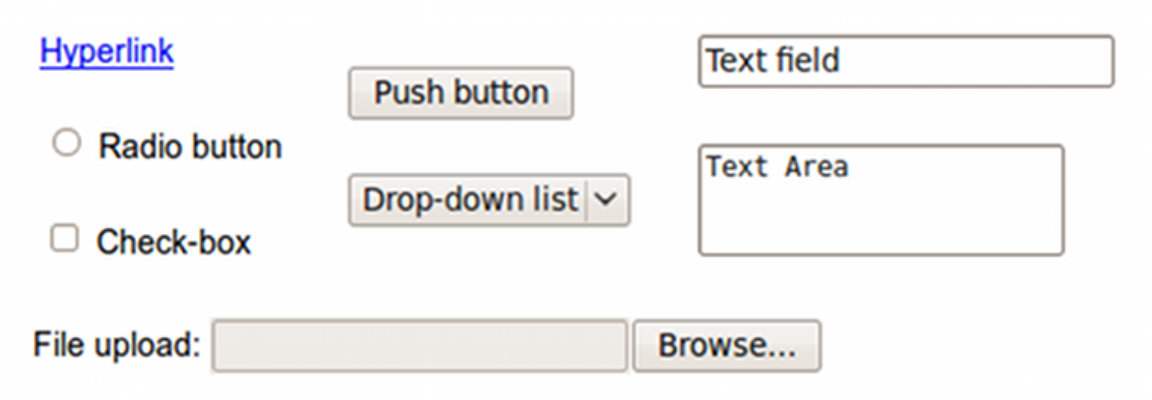

Figure 1: Typical selectable HTML elements

Schrepp and Fischer's study [26] of GOMS [7], a predictive module of interaction, for mouse vs. keyboard web navigation reports a linear relationship between the number of elements in a web page and the selection time for keyboard navigation (using TAB), and a constant relationship for the mouse.

Since the natural flow of TAB-chains is one directional, navigating using the TAB key is laborious. In the worst case – advancing to the previous element – focus advances through every element in the page and all focusable widgets in the browser. While most browsers allow reverse navigation using a modifier key, such as SHIFT, this is not obvious to the user. The reverse TAB-chain may be large too and holding an extra key makes the task even more cumbersome.

This extensive use of the keyboard can potentially cause repetitive strain injury (RSI) [3], especially when combined with ergonomically incorrect use of the keyboard. RSI is not a medical term for an illness, it rather refers to the way an illness is caused and keyboard and mouse are two of the leading causes of RSI in arms and hands. Repeatedly hitting the same key (TAB), and even worse, holding a modifier key while doing it, will likely cause users to bend their wrist. This can cause the Carpal Tunnel syndrome [4], a condition where a nerve running through the wrist is compressed.

Nilsson and Racine [21] found that inexperience is the essential problem with web interaction when a pointing device is not used. They call controls supported by web-based interfaces "simply inappropriate" for users without a mouse, and note that TAB navigation does not provide the same sense of tangibility and directness as a mouse. They considered what a direct-manipulation interface is "intended to do" by revisiting Shneiderman's [28] suggestions to use

- continuous representation of the object of interest,

- physical actions or labeled button presses instead of complex syntax, and
- rapid incremental reversible operations whose impact on the object of interest is immediately visible.

HTML suffices for the first point for keyboard-only interaction. For the second point (avoiding complex syntax), web interfaces suffer because an action as easy as a button press may require a long TAB-chain. For the third point (rapid incremental reversible actions that are immediately visible), use of the keyboard fails because focus transitions from one control to another take many actions. This is especially true when done in reverse.

Clearly, the major issue for keyboard interaction with current browsers is selecting and switching between interactive elements. Not only is a TAB press required for each element, but grouped elements like radio-buttons and check-boxes usually require a TAB press for each element in the group. To make things worse, in some older browsers TAB chains were often required for some inactive elements used only for layout purposes!

A method to directly switch between elements on the same page using the keyboard with minimal key-presses is required. That is the idea behind this thesis.

2.1 Assigning Custom, Logical Indices for TABs

The most basic attempt to improve web navigation using TABs is assigning logical TAB orders to elements based on criteria such as priority of selection. By default, TAB advances

over elements according to their order in the HTML file. This is often different from the order they appear on the web page. Defining a logical TAB order can help and is in fact recommended by the Web Content Accessibility Guidelines [8]. However, this is typically either not done at all or not done correctly. With an incorrect assignment of TAB orders to page elements, during keyboard navigation the focus can jump to a completely unrelated element [17]. This incorrect use of TAB orders has been experienced numerous times by the author of this thesis while working as a web interface developer.

Modifications to a page may result in incorrect TAB orders which is far worse than the longer but correctly ordered default TAB order. This is because these custom TAB orders are defined manually and are hard to manage when modifications are made to a web page. For example, consider assigning custom TAB orders to elements of a web page, starting with menu items within a navigational menu containing a dozen items. Now assume after a modification to the page, a new menu item is added at the very top of the navigational menu. This requires the TAB order assigned to each menu item, and every following item (items with a TAB order larger than the added item), to be increased by one. And if the designers forget to add a custom TAB order to the newly added menu item, the TAB will reach the new element after passing through every single element that has a custom TAB order assigned to it!

When an unexpected element is focused on a page, the user is required to visually scan for the selected element to find the TAB indicator. This visual scan can be slow on a

typical web page with textual links (see the next section). Most web browsers do not provide a clear visual notification of the selected element (e.g., most use a grey dashed border). This adds more visual burden when trying to located a focused element.

2.2 Effect of Visual Scan in Web Navigation

In comparison to desktop applications, in web interfaces the users are more likely to scan for elements because their on-screen location is unknown. In a desktop application users likely won't have to visually scan for the “File” menu, because it is always in the same location on screen. On the other hand, web pages have different designs with different layouts of elements. Even when accessing elements on the same web page, the location of elements can change based on the visible portion of the page when a page cannot entirely fit in a single viewport.

A visual scan to find and distinguish a target element among all the other page elements can take two forms [23] (p. 188). One is the image-matching approach that locates an “image” of the element the user has in mind among the other elements. The other form is the concept-matching approach where an element must be located among a number of visually indistinguishable elements based on the concept. For example, locating a hyperlink within a page full of hyperlinks that all look alike except for their text.

Based on Treisman and Gelade’s feature integration theory [31], in the first approach the users can pre-attentively examine targets (based on shape, colour, etc.) and identify the

target element more rapidly. On the other hand when the target element and the other "distractor" elements look alike, with an increasing number of elements that are displayed simultaneously, the users must rely on a serial scan of all elements to find the desired element.

Web interfaces are more likely to have elements that are visually indistinguishable (such as hyperlinks or text buttons), thus requiring the slower concept-matching approach to find a target element. An efficient web navigation technique must be able to minimize the visual scan as much as possible.

2.3 Programming Better Keyboard Access in Web Applications

Developers of some web-based applications have tried to improve keyboard navigation using client-side scripting to capture keyboard shortcuts. For example, some Google applications including *Gmail* are fully accessible without a pointing device or the need for pressing TAB. This is done by assigning keyboard shorts to all actions, similar to a desktop application. In fact some regularly used actions, such as moving between messages, are easier in *Gmail* than desktop applications because they require a single key press. This method works quite well for these web applications that are used regularly, but does not eliminate the need for a more generic improvement for keyboard navigation that can be applied to any arbitrary web page.

2.4 Mouseless Browsing Extension

While searching for existing research and implementations of keyboard navigation using IDs, a *Firefox* extension, known as *Mouseless Browsing* [22], was found. It was not based on any empirical research on this technique, however it was a relatively good implementation of the technique.

At the time, in the latest edition of this extension, a numeric ID was assigned to each selectable HTML element and a label for this ID was added next to each element within the HTML page. Experiment 1 (Chapter 4) in this thesis is performed using a modified version of this extension.

In order to select an element using this extension, the user had to enter the numeric ID and press ENTER. Alternatively, the selection could be done automatically by a setting a timeout after which the selection was activated.

Screen shots of *Google*'s home page with and without ID labels are shown in Figure 2.

Figure 2: Mouseless Browsing extension (a) without IDs, and (b) with IDs

During initial hands-on experience with this extension, at least two problems became obvious. One was the visual appearance of the ID labels. Labels were added in-line, next to each element causing the elements to slightly move. This changed the layout of the page and was especially noticeable on pages with more tightly packed elements and menus. In some cases, adding the labels caused some elements to fall completely out of position and into the next line.

The other problem was that, given the default settings, the ID labels were shown all the time, causing clutter on the web page. There was an option to toggle visibility of the labels on demand. If enabled, a key had to be pressed to load the labels. This meant

selecting an element using the technique had to be done in two steps: one to load the labels and one to make the selection. This could be considered “complex syntax” and would contradict Shneiderman's suggestions. Additionally, since the labels were added within the page, loading the labels was slow and caused existing elements to move.

2.5 Other Similar Techniques

Spaltholz et al.'s *KeySurf* [30] (see also [29]) is a similar technique that uses labels to facilitate element selection for disabled users. When *KeySurf* is enabled in a browser, labels associated with selectable elements are shown on screen. The elements can be selected by typing the text of the associated label and then pressing ENTER to select the appropriate element.

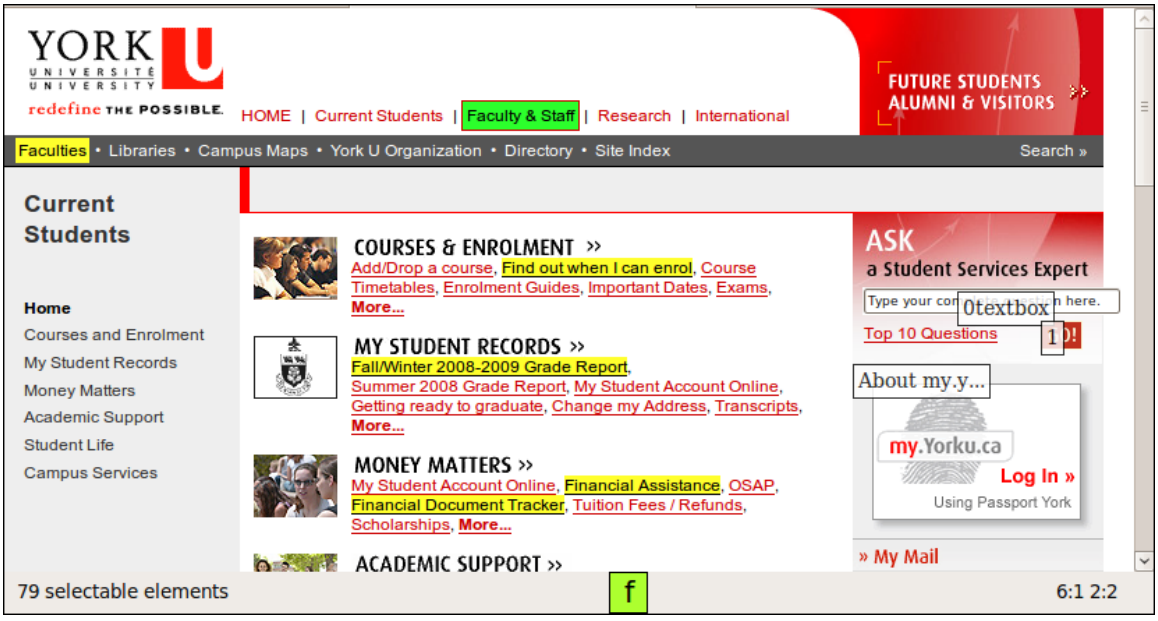

Figure 3: KeySurf (The F-key is pushed.)

In Figure 3, the F-key is pushed down and the targets that start with the letter "F" are highlighted. The most likely candidate is highlighted in green and other possible candidates are highlighted in yellow. There is no need to type the complete text in the labels. Elements with the highest priority (in green) can be selected right away and there are numeric shortcuts for the first few candidates that come next. If none of these candidates is the desired element, the user must continue typing to further limit the set of candidates.

KeySurf uses algorithms to predict which element(s) the user is most likely to select next. While this is beneficial for sites visited regularly, a simple incremental approach to creating labels is more predictable in general.

In a hands-on experiment, *KeySurf* did not go beyond the "Loading..." step on some larger web pages (including the page shown in Figure 3). This could be an implementation-specific issue and not related to the core technique. Another problem with *KeySurf* was representation of unnamed elements. For example, text boxes are identified using "0textbox" and "1textbox". In case of images, if there is an alternate text within the image, the alternate text is used as the label. While this is practical, it is unpredictable. *KeySurf* continuously shows the labels on screen and this can cause clutter. Having labels with text as large as "0textbox" can significantly add to the clutter. Another important limitation of *KeySurf* is that it relies on all letters of the alphabet, as well as other keys (e.g., ESCAPE or ENTER), rather than a limited set of characters or keys.

KeySurf is designed for the disabled and so it is optimized for this specific group of users. The goal of this thesis is to present a generic improvement to keyboard navigation that can benefit different demographics of users, for example all those who prefer (or can benefit from) keyboard navigation.

CHAPTER 3
ID Navigation

In this chapter, the keyboard navigation technique presented in this thesis is explained in more detail and different variations of the technique are discussed. Later, a predictive model is given to compare the new technique with the existing TAB-key navigation as well as the mouse.

The idea behind keyboard web navigation, henceforth "ID navigation", is to assign a unique, incremental ID to each interactive web page element. Since the early days of graphical user interfaces, desktop applications have used mnemonics [14]. These are underlined characters, normally in application menus, that can be used in combination with a modifier key (e.g., ALT) to select a command or navigational item. ID navigation extends the idea of mnemonics to an arbitrary number of elements by using incremental IDs instead of pre-defined underlined characters. The ID is displayed as a tiny label next to each element, thus only a single visual scan is required to find both the element itself and the ID. To minimize screen real estate and avoid clutter, the ID pops up only when a modifier key is pressed and held.

Initially, we assume keyboard focus is on the actual web page, not on the browser, and that the keyboard is either unused or not used for text entry. For example, arrow keys may scroll the page or change a value in a drop down list, or for example an action key

may toggle a check box; but there is no numeric or text input. Thus, numeric key sequences are available to navigate between widgets based on assigned IDs.

Assigning IDs as incremental digits (instead of using an element's name or probability of selection) allows users to scan the labels in the same top-to-bottom, left-to-right direction as text on a page. For a typical web site, elements at the top of the page are used for navigation and are generally repeated on all pages throughout the web site. Since these are the first elements to have IDs assigned to them, they get short, easier-to-remember IDs and are unchanged across all pages. This makes it easier for users to memorize them for future visits. Where the elements do not appear in the same order as in the HTML file, IDs are still assigned to related elements (such as list items) in sequence.

ID navigation is meant to improve the existing keyboard navigation technique, not replace it. It is to facilitate faster movement between elements that are too far apart for efficient TAB access. For example, the user can use TAB navigation when filling out a web form's inputs one by one and use ID navigation to "jump" to a different part of the page.

3.1 Numeric ID Navigation

With the original *Firefox* extension (*Mouseless Browsing* – Chapter 2.4), typing a numeric ID followed by a short timeout or an action key switched focus to the target element. In Figure 4, a hyperlink with numeric ID of "7" is selected using the extension. This method requires 2-3 keystrokes for selection: 1 or 2 numeric key presses for the ID

and 1 key press (or a timeout) for selection. Clearly, this is less burdensome than traditional TAB-chain navigation.

Figure 4: Element selection using "Mouseless Browsing" extension

A simple case study can illustrate the difference between TAB-key navigation and ID navigation. Consider selecting the radio button “pages from Canada” (assigned ID “5”) in Figure 4. By default, when this particular Google home page is loaded, the search box is focused. Using TAB-key navigation, seven TAB presses are required to reach the radio button. The TAB indicator passes through the buttons and hyperlinks on the right side of the search box before reaching the target radio button. Then, the user must push another key to make the selection. On the other hand, using ID navigation only a single key press (“5”) is required to directly select the element. A second key may be needed to make the selection using ID navigation if the timeout option is not used.

Natural numbers (i.e., positive integers) are especially useful when a numeric keypad is accessible to the user. There users (e.g., accountants, bank tellers) who are experts in

number entry using the numeric keypad. Using ID Navigation on the numeric keypad can potentially benefit this group of users as well.

Additionally, using natural numbers allows the technique to easily extend to speech recognition systems. With a limited vocabulary of only 10 digits, the chance of error during speech recognition is low. Due to the importance in recognition of spoken numbers (e.g., phone numbers, postal codes) in automated telephone systems, there are algorithms designed specifically for recognition of natural numbers [10]. So, even when the numbers are not read one digit at a time, it is easy to find a simple speech recognition system to detect the spoken numbers and implement it alongside ID navigation.

The main problem with the original extension was that the IDs were either shown continuously, which caused clutter (see Figure 5), or were "toggled" using a key. If the extension was activated, users had to hit a toggle key to bring up the IDs, and then perform the selection. This was slow and increased the visual and cognitive burden. Thus, a modification was required to show the IDs on demand without requiring "complex syntax".

Figure 5: Portions of a relatively cluttered page with and without ID labels

3.2 Ambiguity with Text-input Controls

Text-input[1] elements pose a challenge for ID navigation. When focus is on a text-input element, a method is required to distinguish between the control ID and the data entered into the element. Two solutions to this problem were provided by the original *Mouseless Browsing* extension.

One solution was to use the numeric keypad exclusively for navigation and the main number keys for non-navigational number entry. This is a problem for users who prefer the numeric keypad for entering numbers, and for those who use a laptop. Numeric keypads are awkward to use on notebook computers because they are integrated in the

[1]For keyboard navigation, drop-down lists are like text-input elements because key sequences are used to scan through available values. This is useful for keyboard navigation, especially for larger drop-down lists.

main keyboard and are accessible only through a mode shift, which requires an extra keystroke. Additionally, this method only works when IDs are numeric.

An alternative was to use a modifier key in text-input elements to process the numeric keys for navigation. In the original extension, the CONTROL key was assigned to toggle numeric entry. The technique was modified to show the IDs only while this "activation key" was held, allowing users to hold the modifier key while entering an ID regardless of the past state or the keyboard focus. This results in a more consistent interaction, eliminates the ambiguity with text-input controls and avoids clutter by showing the IDs on demand, all with a single keystroke.

3.3 Beyond Numeric IDs

While numbers are incremental and easy to follow, the IDs can use any character in the keyboard. This is particularly useful if a numeric keypad is not available, as for laptops.

Using ID navigation with a limited number of keys can greatly benefit disabled users. There are users with neuromuscular impairments who cannot use fine movements with a mouse (or use a mouse at all) but can use a keyboard (see section 1.1). Some group of users can only use a limited set of keys rather than a full keyboard. Special input devices such as chord keyboards [9] (p. 59) designed for this group of users can be used with ID navigation to facilitate web navigation for them.

A 4-finger variation of ID navigation where four fingers are used on four adjacent keys is evaluated in Chapter 6. With enough practice this method has great potential for touch typists. For this variation, a candidate group of keys is J, K, L and semicolon (;) as these keys form the normal home position for the right hand when touch typing. Also, the J key has tactile feedback, making it easier to put the fingers back in position. A small pilot study of this variation showed using the symbols J, K, L and ";" in labels was confusing, so numbers 1-4 were used on the labels instead. The keys were also labeled with these numbers to avoid confusion, however a small between-subjects study showed no improvement in selection time when labeling the keys. See Figure 6.

Figure 6: ID navigation with four fingers

A similar group of keys would be A, S, D and F using the left hand. This allows full navigation with the left hand alone by pressing the activation key with the left thumb. Such left-handed variation of 4-finger ID navigation allows navigation with only the left

hand, thus reducing the load on the right hand. However, not all users may comfortably hold the activation key (e.g., ALT) with the left thumb while typing the labels with the other four fingers of the same hand.

3.4 Predicting Using a Keystroke-Level Model

The selection time to move from one widget to another for the techniques discussed above can be predicted using Card et al.'s Keystroke-Level Model (KLM) [6]. In its general form, the model gives

$$T_{Execute} = nT_K + nT_P + nT_H + nT_D + nT_M + nT_R \quad (1)$$

where the total execution time is predicted by summing primitive operations.

- T_K is the time for a keystroke or button press and is calculated for any key including mouse buttons and modifier keys.
- T_P is the time to point to a target with the mouse.
- T_H is the homing time for the hand to switch between the keyboard and the mouse.
- T_D is the drawing time (unused in this thesis).
- T_M is the mental preparation time.
- T_R is the system response time.
- n is the number of repetitions of each operator.

To simplify the calculations and consider the web interface alone, we can assume a page fits in a single view port and no scrolling is needed. Given this, the total number of page elements is the same as the number of visible elements.

Predictions are developed below for a single selection of a focusable element on a web page using three different navigation methods.

3.4.1 Using the Mouse

$$T_{Execute} = T_H + T_M + T_P + T_K \quad (2)$$

where

- T_H is the homing time. It equals 0.4 s [6] for switching from text-input widgets and 0 for other widgets where there is no need to use the keyboard.
- T_M is the mental preparation time to execute pointing. T_M is set to 1.35 s for all devices, as given by Card et al. [6].
- T_P is the pointing time to select the new widget with the mouse. It ranges from 0.8 to 1.5 s [6].
- T_K is the time to press the mouse button. Based on Card et al. [6], T_K ranges from 0.08 to 1.20 s; however, since this range was estimated from participants' typing speed rather than a mouse click, it makes more sense to use the lower end of 0.08 s.

Given this, the estimated execution time to switch between widgets using the mouse is between 2.23 and 3.33 s.

3.4.2 Using the Keyboard: Tab Chains

$$T_{Execute} = T_M + n_{TAB}\, T_K + T_M + T_K \quad (3)$$

where

- T_M is the mental preparation time, as explained above. Based on heuristics for placing M operations [6], T_M is calculated once before the first TAB and once before the final keystroke required to select the desired element.
- n_{TAB} is the number of TABs to reach the target element. It is between 1 and n where n is the number of TAB-sensitive interactive elements. For a page with n elements, the average number of TABs is:

  $$n_{TAB} = (n + 1) / 2 \quad (4)$$

- The first T_K in eq. (3) is the time to press TAB. It ranges from 0.08 to 1.20 s. Arguably, users who prefer keyboard navigation have above-average typing speeds, and since all strokes are on the same key, a lower T_K than given by Card et al. [6] may be warranted.
- The second T_K is the keystroke (typically ENTER or SPACE) to select the element once focused. It is included for elements that need selection, like buttons and hyperlinks.

Given the above model components, the estimated time to switch between elements using TAB-key navigation is

$$T_{Execute} = \alpha + \beta n_{TAB} \text{ (s)} \quad (5)$$

where α ranges from 2.78 to 3.9, β equals the typing speed and ranges from 0.08 to 1.2, and n_{TAB} is estimated by the average number of elements on the page using equation 4.

The model assumes elements are traversed in the normal order and does not consider reverse traversal using SHIFT.

3.4.3 Using the Keyboard: ID Navigation (All Variations)

$$T_{Execute} = T_M + n_K T_K + T_R \quad (6)$$

where

- T_M is the mental preparation time to prepare to enter the ID of the element, as described above.
- n_K is the number of keystrokes to enter the ID and execute the action. Assuming a 2-digit ID, two keystrokes are needed to enter the ID and if enabled, a third key (e.g., ENTER) to confirm selection. Plus, there is an additional keystroke for the activation key, if enabled.
- T_K is the keystroke time, as explained for TAB-chain navigation.
- T_R is the response time. If a timeout is used instead of a key press, T_R = 0.5 s, otherwise zero is used. 0.5 s is the timeout value used in Experiment 1 (Chapter 4). In practice, if the timeout is used, it should be user configurable.

The estimated time to switch between elements using ID navigation is between 1.59 to 4.95 s using the action key and between 2.09 to 5.45 s using a timeout of 0.5 s (when an activation key is used).

3.4.4 Comparing the Estimated Execution Times

Table 1 summarizes the model predictions for each navigation method. While model constants from Card et al. [6] may be inaccurate, the importance of the predictions lies in the relative comparisons they afford. The selection time for TAB navigation is a linear function of the number of TAB-sensitive elements. For both mouse and ID navigation, $T_{Execute}$ is constant regardless of the number of elements.

Table 1: Summary of Model Predictions (s) by Method

Method	Model Prediction (s)	
	Min	Max
Mouse	2.23	3.33
Tab		
5 elements	3.18	9.90
30 elements	5.18	39.90
60 elements	7.58	75.90
ID Navigation		
key selection	1.59	4.95
0.5 s timeout	2.09	5.45

These results are consistent with Schrepp and Fischer's predictive comparison [26] of mouse vs. keyboard web navigation (see Chapter 2).

Furthermore, web-authoring statistics by WebWatch [16] and Google [2] show an average of 60 elements per web page and at least 20 elements for over 97% of web pages. Thus, the predictions for TAB-chain navigation in 3.4.2 may be on the low side.

The relationship between the average number of keystrokes and the number of web page elements is illustrated in Figure 7. The number of keystrokes for TAB navigation technique is clearly a linear function of the number of elements while the relationship is logarithmic for ID navigation techniques. While the 100,000 elements are only included in the graph for better illustration of the relationship, it is common to see pages with 100 or even 1000 interactive elements. Even on a page of this size, the techniques differ by a factor of ten or even more.

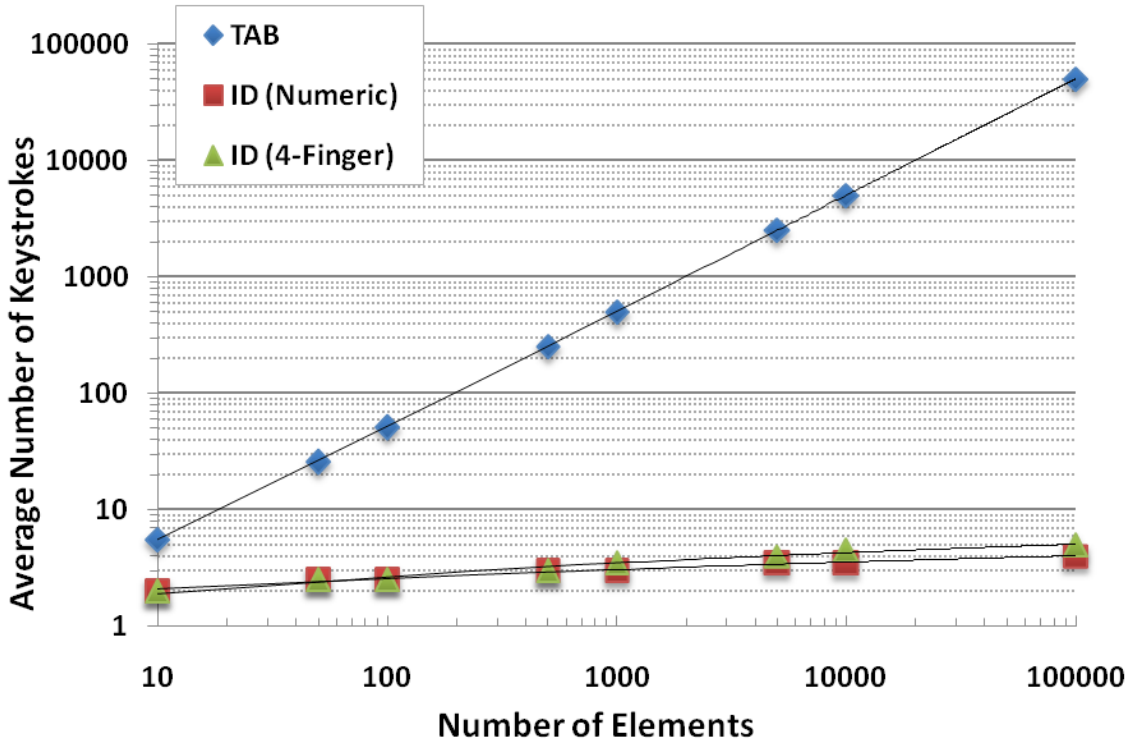

Figure 7: Average number of keystrokes vs. number of page elements

The predicted results for the mouse and ID navigation techniques are similar and both are predicted to be much faster than TAB navigation. A sensitivity analysis of the predicted selection times based on keystroke time (T_k) of the users, for a page with 200 elements is shown in Figure 8. The crossover point is at a keystroke time of 0.60 seconds, which matches a typing speed slower than the average non-secretary typist [6]. This means, for users with above average typing speeds ID navigation is predicted to be faster than the mouse.

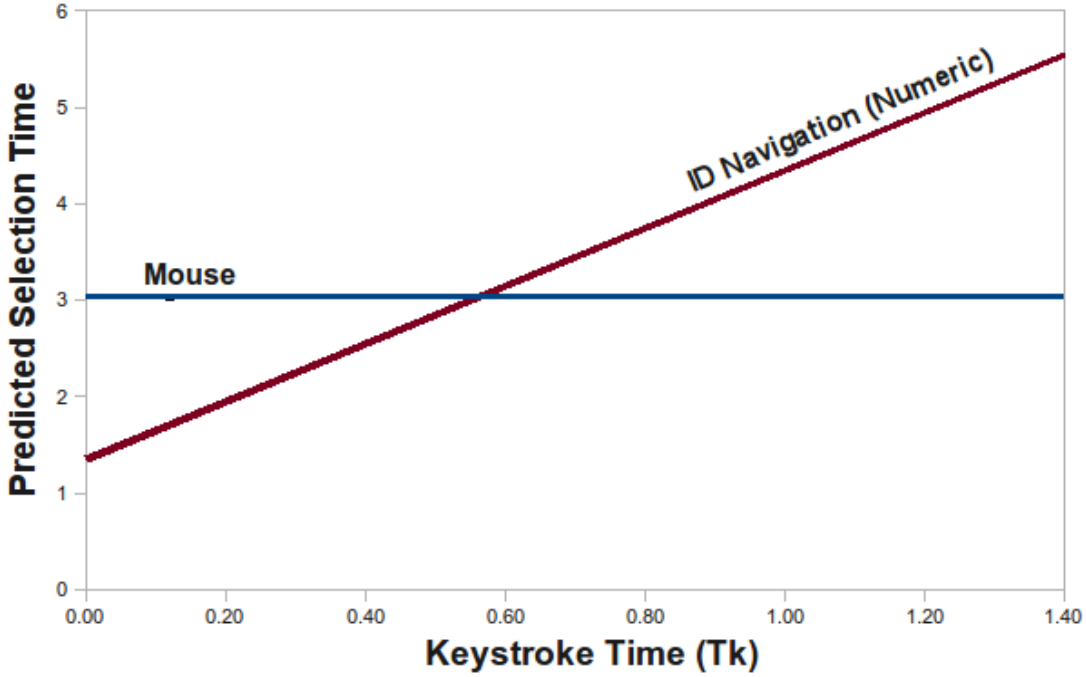

Figure 8: Sensitivity analysis of the predictions based on T_k

While these results look optimistic, an empirical comparison of the techniques is needed to test the predictions.

CHAPTER 4
Experiment 1: An Initial Evaluation

4.1 Evaluation Method

A simple experiment was designed to compare the time to select a random target button from a set of buttons on a web page using the three interaction techniques described above. The experiment was limited to a fixed view port with no text-input elements. The goal was to perform a preliminary evaluation to test the model's predictions from the previous chapter and to inform subsequent design.

4.1.1 Participants

Eleven participants (6 male, 5 female) performed the experiment. Most were in their 20s and all were experienced computer users. Participants were students of the local department and were not paid for this experiment. In a questionnaire given to them before the experiment, 8 participants mentioned Mozilla *Firefox* as their default browser and the other 3 had used *Firefox* in addition to their default browser.

Nine participants said they normally used keyboard shortcuts in the browser in addition to the mouse. More than half the participants said they only used the keyboard for the browser's own features (opening tabs, focusing on the address bar, etc.) and the rest said that they preferred the keyboard to the mouse because it was faster.

4.1.2 Apparatus

The experiment was performed on a Linux-based machine in the local computer lab. Mozilla *Firefox* was used because it provides good keyboard support to access features, and is easily modified and expanded using extensions or add-ons.

As explained in section 2.4, in the original *Mouseless Browsing* extension IDs were either always shown or toggled with a key and this caused the web page to be cluttered with lots of IDs. Thus the extension was modified it to show the IDs on demand while an activation key was pressed. The computer was fast enough to show the IDs without delay. The original extension used the CONTROL key to send keystrokes to the navigation processor when focus was on a keyboard sensitive element; thus, the same modifier key was used as the activation key. The IDs were shown until either the CONTROL key was released or a selection was made.

Several preliminary tests examined different methods of performing selection after the IDs were entered. Initially, hitting ENTER after the IDs were entered selected the appropriate element. However, this introduced errors since participants could accidentally hit ENTER twice and select a second unknown element. In the final modification used in the experiment, a 500 ms timeout following the last keystroke selected the appropriate element. It is best to allow the users to adjust the timeout, however, for the purpose of the experiment the same timeout value was used for all participants.

All three input techniques were tested on a custom interactive web page written in Javascript and stored on the local computer to avoid communication delays. The web page consisted of a start button, a message box for instructions and feedback, and 30 target buttons.

To test the model's predictions, the elements were labeled and ordered to minimize the visual scan time for all input techniques. The buttons were labeled with numbers corresponding to their IDs (and their TAB order). Since the ID for the start button was "1", the target buttons were numbered from 2 to 31 to match their ID for consistency ("Button 2", "Button 3", etc).

Figure 9 shows the experiment web page in a situation where CONTROL is held and the IDs are shown. As shown, the CONTROL key is pressed causing a numeric ID to appear beside each focusable web page element.

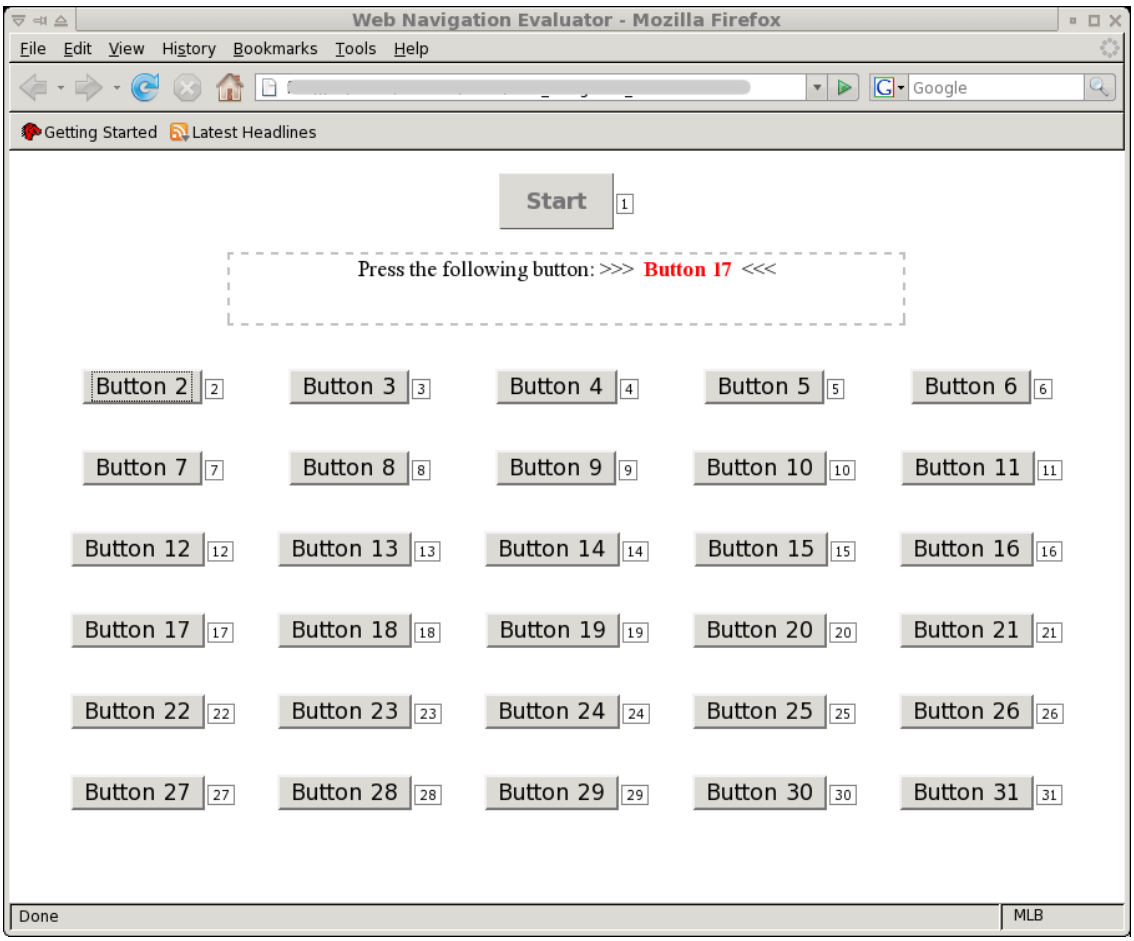

Figure 9: The experimental interface during a trial (Experiment 1).

Initially, only the start button was enabled on the page. Once participants clicked the start button, it was disabled and the target buttons were enabled. This also started a timer. Once a target button was selected, the timer stopped and the message box indicated if the correct button was pressed and instructed participants to start the next trial when ready. For each trial, data were logged indicating the target button, the button actually pressed, and the selection time.

4.1.3 Procedure

Before gathering data, the ID navigation technique was explained to participants. They were allowed a few minutes to practice the technique on a random web site. A couple of participants who weren't familiar with TAB-chain navigation practiced that method as well. Once a participant was comfortable with the system and the timeout threshold, data collection began.

Participants were instructed to press the start button when ready. Upon pressing start, a target button was indicated below the start button (shown in Figure 9). They were instructed to proceed as quickly and accurately as possible to the indicated button and select it using the navigation technique under test. They were told that once a target button was pressed (and the message box confirmed the action), they could stop and relax before pressing start again. If an incorrect button was selected, this was indicated in the message box; however, participants continued without redoing the trial.

To keep TAB-chains inside the web page, focus was initially on the start button. The start button was disabled after it was pressed, with focus advancing to the first target button (Button 2). As this is slightly different from normal TAB behaviour of a browser, it was explained to participants. Thus, they did not need to visually scan for the focus indicator every time.

4.1.4 Design

The experiment was a 3 × 15 within-subjects design. The factors and levels were as follows:

Input Technique	ID navigation, TAB navigation, Mouse
Trial	15 buttons (#3, #5, #7, …, #31) in random order

The dependent variables were selection time per button (s) and error rate (%). The total number of trials was 11 participants × 3 input techniques × 15 trials = 495.

To minimize skill transfer, the input techniques were assigned to participants using a Latin square, with each order assigned to either two or three participants[2].

4.2 Results and Discussion

The grand means for the dependent variables were 2.96 s for selection time and 4.63% for error rate. The effect of interaction technique on these measures is reported next followed by the effect of the TAB order and row of buttons. Finally, the observations are compared with the model predictions.

[2] The conditions were balanced using a Latin square for a factor of 4. However, since this initial experiment was tested on only 11 participants (instead of 12) it was not fully balanced.

4.2.1 Selection Time

As expected, TAB navigation was the slowest technique with a mean selection time of 4.44 s. The selection time was 42% less for ID navigation (2.56 s) and 57% less for the mouse (1.90 s). The differences were statistically significant ($F_{2,10} = 42.26$, $p < .0001$). See Figure 10.

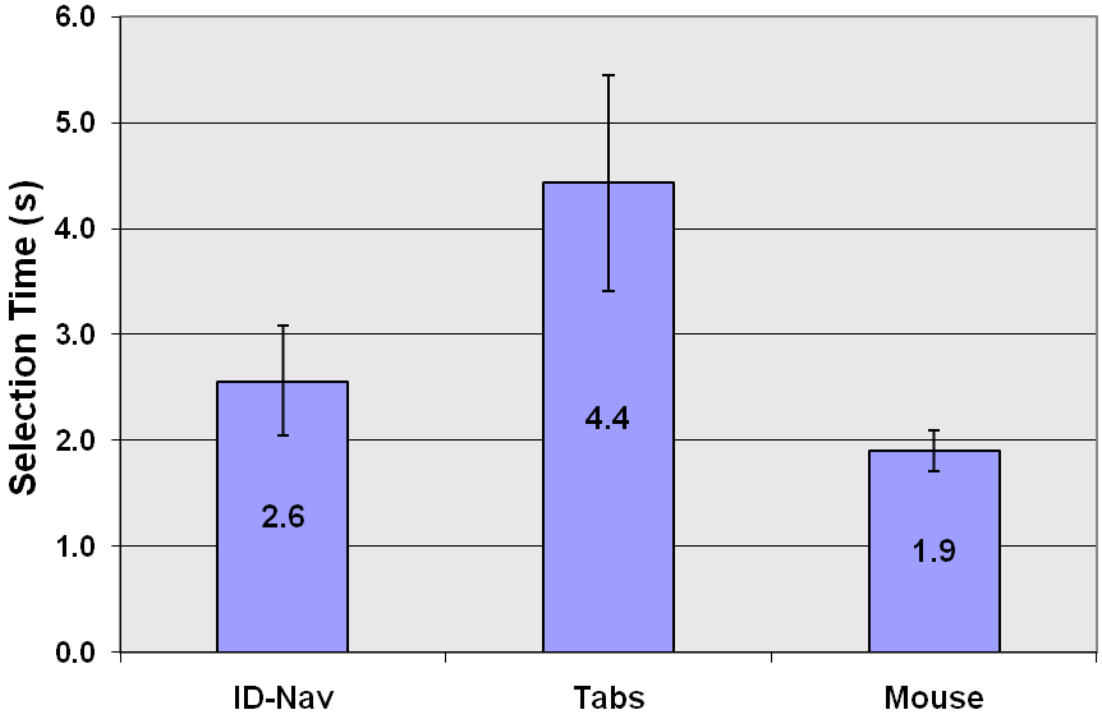

Figure 10: Mean selection time (s) by input technique (Experiment 1)

(Note: error bars show ±1 standard deviation)

Based on selection time, ID navigation is clearly a good alternative to TABS. Since the number of elements in the test web page was about half the estimated average for a normal web page [2,16], the difference may be even higher for real web pages. The

difference between the mouse and ID navigation was smaller than the difference between each of them and TABS.

Notably, this experiment served mainly to test the predictive model and compare the two keyboard techniques. The experiment did not consider homing time for the mouse in web pages where typing is needed (see Experiment 2). Taking homing time into account, ID navigation may even compete with the mouse for expert keyboard users. See the subjective responses of participants below for more discussion.

Order of the Buttons

The effect of the order of the buttons (their TAB order on the page) on the selection time was analysed for each input technique. Figure 11 shows the relationship between selection time and the order of the buttons. The linear relationship for TAB chains is clearly seen. The regression lines show a relatively flat relationship for ID navigation and mouse conditions.

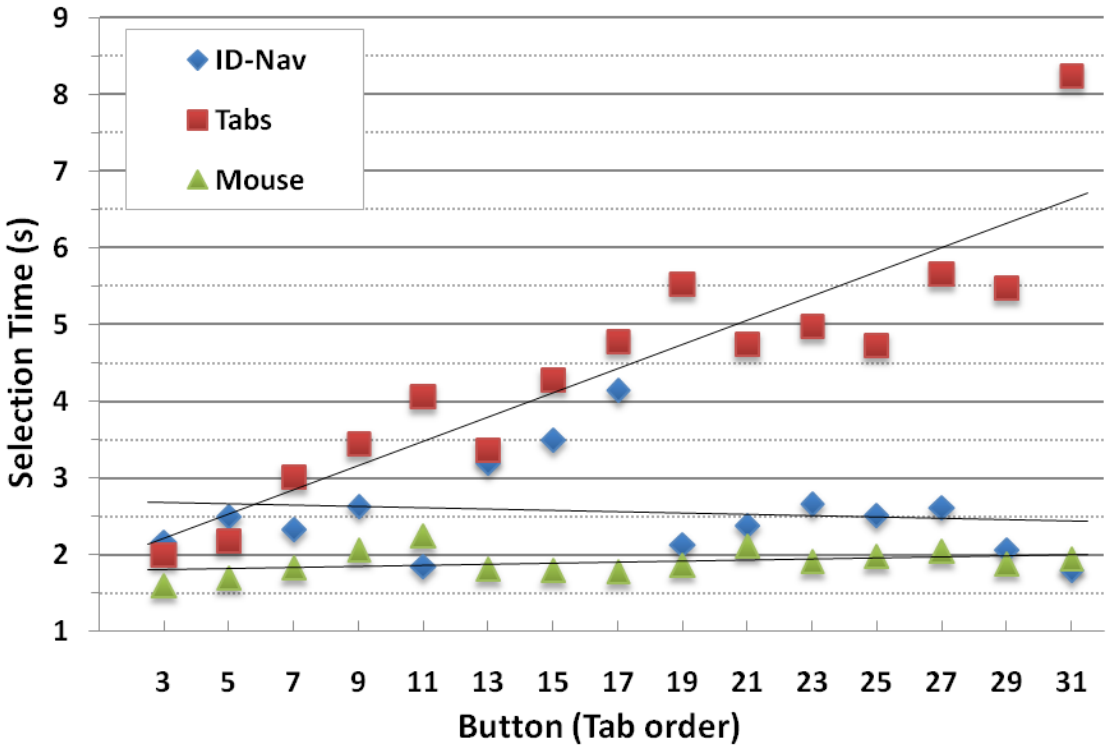

Figure 11: Selection time by button (tab order) and input technique (Experiment 1)

An interesting observation for TAB navigation was the strategy by some participants to press and hold the TAB key until they almost reached the target button. This didn't seem to help as in many cases they missed the target and had to traverse the keys in reverse order by holding SHIFT. This reverse traversal was unnatural and seemed to take much longer than the normal traversal. Some participants even had to visually scan for the SHIFT key when this occurred.

Importantly, TAB order is not the same as pointing distance for the mouse condition. For example, to select buttons on the first row, the button in the middle was closest to the start button (see Figure 9), yet it had a higher TAB order than the first button on that same

row. To make sure the difference between target distance and TAB order would not affect the results for the mouse in this experiment, the same analysis was performed on relative position of the buttons based on selection time per button row. The results show a similar relationship to TAB orders with similar statistical significance for all three techniques (see Figure 12). The pointing distance for the mouse, while having a tiny upward slope, was too small to significantly affect the selection time.

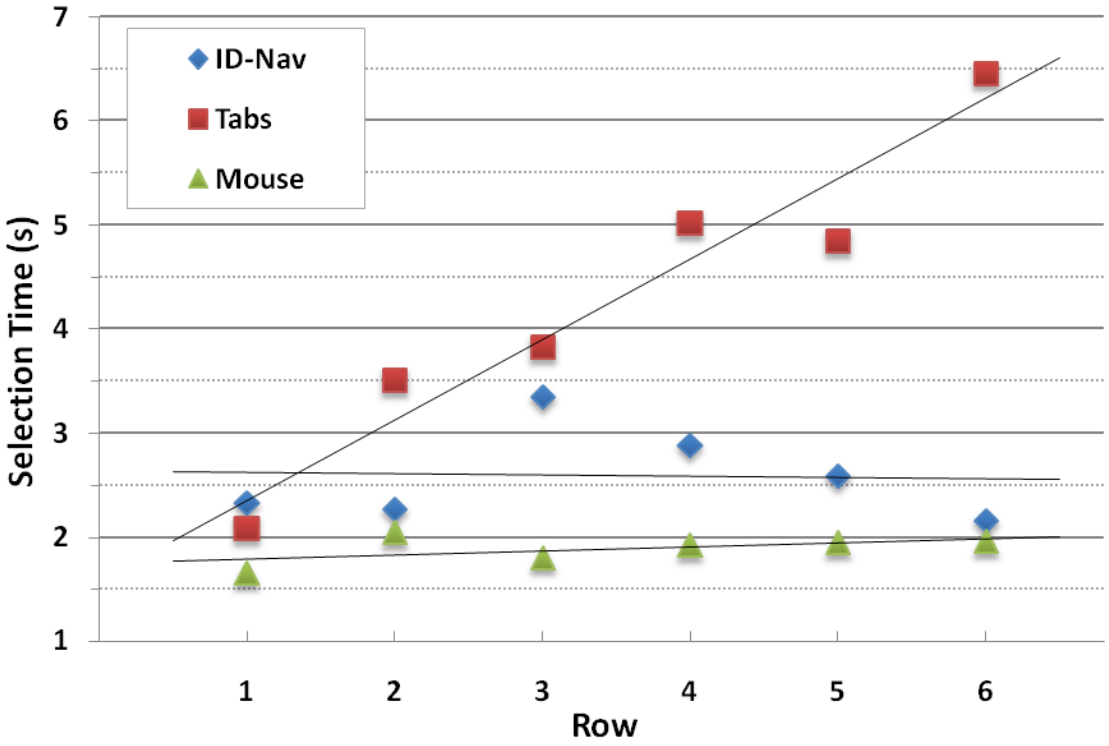

Figure 12: Selection time (s) by button row and input technique (Experiment 1)

Comparing the Observations with the Predictions

The observations fall close to the lower end of the predictions from the keystroke-level model (see Table 2).

Table 2: Model Predictions vs. Observations (Experiment 1)

Method	Model Prediction (s)		Observations (s)
	Min	Max	
Mouse	2.23	3.33	1.9
Tab chains			
30 elements	5.18	39.90	4.4
ID Navigation			
0.5 s timeout selection	2.09	5.45	2.6

For the mouse, the mean per-button selection time from the experiment was 15% less than the minimum predicted selection time. One explanation is that because the buttons were relatively large they reduced the pointing time according to Fitts' Law [20], while this is not accounted for in the pointing operator in Card et al.'s [6] original model.

For both keyboard navigation techniques, the observations fall close to the range of the predictions. The mean selection time for ID navigation was 24% larger than the minimum estimated time. The observed mean selection time for TABs was only 15% smaller than the minimum predicted selection time for a page with 30 elements. Based on this comparison TABs were typed relatively fast during this experiment, thus for users with disabilities or with poor keyboarding skills, the selection time for TAB navigation might be even larger.

Use of a Numeric Keypad for ID Navigation

To estimate usage of the numeric keypad, participants were instructed to use their preferred entry method for ID navigation. Only three participants did not use the numeric keypad. Two of these were laptop computer users. Based on these results, use of the numeric keypad exclusively for ID navigation is not recommended in general, since it is already the preferred choice of numeric entry for many users.

4.2.2 Error Rate

Mouse navigation was the most accurate technique with an error rate of only 0.56%. The error rate was higher using TABS (3.3%) and ID navigation (10.0%). See Figure 13. The differences were statistically significant ($F_{2,10} = 5.63$, $p < .05$).

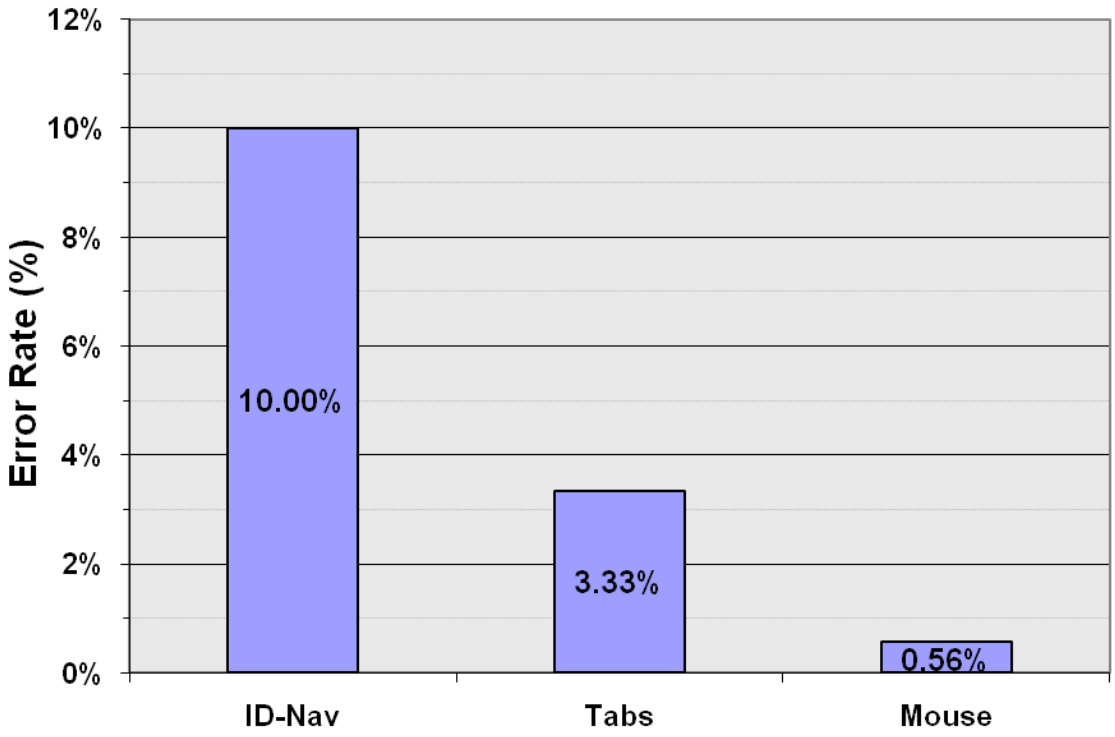

Figure 13: Error rate (%) by input technique (Experiment 1)

An error was logged when the wrong button was selected. For the mouse, clicking outside a button (i.e., in the space between buttons) was not recorded. This was observed with reasonable frequency; thus, the error rate for the mouse is artificially low. On most real web pages, elements are typically smaller and closer together; so the true error rate for the mouse would likely be higher. On the other hand, having small, tightly packed elements should not affect the keyboard navigation techniques.

The 10% error rate for ID navigation in this experiment was a concern. One cause, no doubt, was participants' lack of practice with this new technique. However, inspection of the data along with the observations by the experimenter suggested two additional causes.

First, the 0.5 s timeout might have been a bit short. Participants sometimes paused too long between the 1st and 2nd digits while entering the ID and this caused a timeout and inadvertent entry of the 1st digit only. These errors are shown in Figure 14 with the label "2-digit ID, 2nd digit dropped". They constitute 44% of the 10% ID navigation errors.

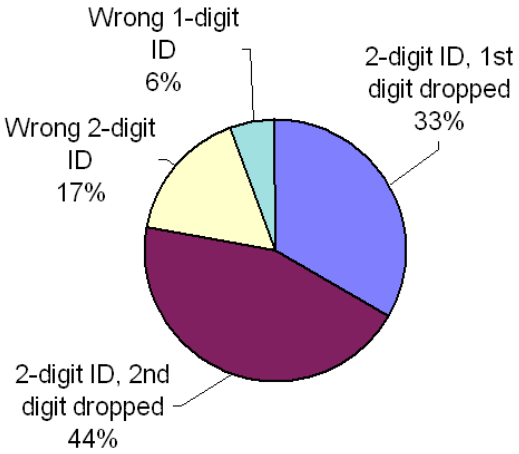

Figure 14: Breakdown of the 10% errors for ID navigation (Experiment 1)

Second, participants occasionally pressed the CONTROL key slightly after entering the 1st digit and this caused inadvertent entry of the 2nd digit only. These errors appear with the label "2-digit ID, 1st digit dropped". They constitute 33% of the 10% ID navigation errors. This behaviour was likely due to the interface using button labels containing the ID (e.g., "Button 23" having ID = 23; see Figure 9). This may have caused a slight tendency for participants to "jump the gun" since they "knew" what ID to enter. An alternative experiment design could assign random ID numbers to web page elements, thus forcing participants to pause slightly (~200 ms [15]) to perceive the ID number before entering it.

Notably, these two types of errors are artefacts of the experiment or implementation, not inherent limitations in ID navigation.

Concerning errors due to the short timeout threshold, this could be mitigated by choosing a longer threshold, say 0.75 s. Further analysis showed that participants with the most errors also indicated they were not heavy keyboard users. Thus, ID navigation might improve using a configurable or adaptive timeout, where longer timeouts are set based an individual's keyboard usage or typing speed. While in the experiment we had to select the same timeout threshold for all participants, the technique does allow the timeout to be easily configured by the users. As ID navigation is a new technique, continued practice would likely help as well.

To avoid errors caused by pressing the CONTROL key late or releasing it too soon, one participant suggested the modifier key should "stick". However, a modifier is still needed to send keystrokes to the navigation processor when a text-input element has the keyboard focus. So, for consistency it is best to use the same method regardless of the previous element.

4.2.3 Questionnaire

Participants were given a second questionnaire after the experiment. On a 7-point Likert scale of (1 = not comfortable at all, 4 = somehow comfortable, 7 = very comfortable), they were asked to rank their level of comfort with ID navigation. They responded with

an average score of 5.4. They were asked if they would enable ID navigation on their own browser. On a similar scale, the average score was 4.9.

Additionally, participants were asked to rank their preference of the three input techniques from 1 to 3 (1 = most preferred). ID navigation had the highest preference (even above the mouse!) and TAB navigation had the lowest preference. Participants found TAB navigation too time consuming to be practical.

Participants were asked if they used web-based data entry in their field of work or study. Four indicated they did. Of these, three indicated they use the keyboard for navigation. The others showed considerable interest in using ID navigation in the future, having participated in the experiment.

Participants were also asked for general suggestions to improve ID navigation. Most commented that the time threshold needed to increase. One participant suggested that when IDs reached three-digit numbers, letters could be used instead. Some users suggested the visual feedback for focus needed to be more visible than the default dashed line; however, this was more of a concern for TABs rather than for ID navigation.

While the implementation was fast enough for the web page used during the experiment, some participants noticed a delay when the IDs were being loaded on larger web sites.

Chapter 5
Improving ID Navigation

This chapter summarizes the problems with the original implementation based on the results from Experiment 1 followed by the improvements made to a new implementation of ID navigation. Afterwards, the time to perform a task involving data entry is predicted using the KLM model. The improved implementation is used in Experiment 2 (detailed in Chapter 6) which includes an empirical study of a task involving data entry.

5.1 Problems with the Original Implementation

Below is a list of problems with the original implementation used in Experiment 1 (Chapter 4) as well as a few previously known problems that were not related to Experiment 1, but needed to be fixed before such implementation could be used in real life or in a more comprehensive experiment.

- The most important problem with the original implementation of ID navigation was inaccuracy, with an error rate of 10% in Experiment 1. Based on a breakdown of the errors made using ID navigation in Experiment 1 (see Figure 14), most errors were likely caused by the timeout used to make selections. Thus, an improved implementation of ID navigation would require a better method of selection.

- As mentioned earlier, in the original extension the ID labels were presented in line with HTML elements. This would cause clutter (see Figure 5) and slow down presentation of ID labels on demand. This remained unchanged for Experint 1 as it wouldn't affect it, but could turn out to be a significant problem if ID navigation was used on larger web pages with more elements.

- A method was needed to make corrections by canceling selection of a target element if a mistake was made.

- An improved highlighting could help users find the target element faster, especially on a web page with many smaller elements.

5.2 Improving the Implementation of ID Navigation

A new extension was designed for Experiment 2 (Chapter 6) with significant improvements based on the problems mentioned in the previous section as well as user feedback received through small pilot studies during the reimplementation process:

- In the new extension, the target element was selected as soon as the activation key was released. Selection after a timeout delay caused high error rates in Experiment 1 and selection using the ENTER key was potentially error prone (see section 4.1.2 for details). Both latter methods were implemented in the new extension as they could potentially become useful. For example, if an alternate method of entering IDs such as speech recognition was used, a confirmation

would be helpful before a selection was made. Nevertheless, by default, the target element was selected by releasing the activation key. Elements such as hyperlinks or buttons were clicked on selection.

- To improve the response time and minimize clutter, the IDs were presented in a different "layer" on top of the original web page. This method didn't change the original layout of the page in any way. It was also fast-loading since the original layout didn't need to be modified and the existing elements didn't need to be moved. The IDs popped up when an activation key (ALT) was pushed down, the characters were entered while it was being held down and the target element was selected as soon as the activation key was released. The activation key was changed from CONTROL to ALT to minimize conflict with keyboard shortcuts already assigned in *Firefox*.

- To avoid mistakes, a delete key was introduced to clear the current selection before releasing the activation key. After hitting the delete key, the users could release the activation key without making a selection, or type in the ID of a new element without having to release the activation key.

- An improved highlighting was introduced by using a box with a transparent yellow background and a red border instead of the default grey dashed border used in *Firefox*. This highlighting was only shown *during* selection to differentiate between an active selection and a previously selected element. As

soon as the selection was confirmed (by releasing the activation key), the extra highlighting disappeared and the selected element was focused with the default dashed border.

- The new implementation supports a wider variety of HTML elements (such as image maps [24]), provides better support for dynamic web pages and improves label creation for certain representation of HTML elements such as elements with "fixed" positioning and invisible elements. It also provides a better visual appearance for labels than the original implementation.

A list of other improvements made to the new implementation of ID navigation is presented in Appendix A: Implementation Changelog.

5.3 Predicting Data Entry Time

A similar prediction using the Keystroke-Level Model is given for a unit conversion task involving data entry. In this task, the user must select the type of conversion (distance, weight or volume), enter the value to convert, select the units from two consecutive and similar drop-down menus (for the source and target units) and press a convert button. This task is studied in Experiment 2 (Chapter 6) and the web page used in the experiment is shown in Figure 15b. In this web page, a number of random hyperlinks were added in two parts of the form to require participants to "jump" to a different section. Without these links the task would be filling out a normal web form where TAB-navigation is

expected to perform best, however the purpose of this thesis is finding a faster keyboard navigation technique for when TAB-navigation fails.

As with the original prediction, we assume the web page fits in a single view port and no scrolling is needed. This assumption is true for the web page used in Experiment 2.

Predictions are developed below for the unit conversion task explained above using three different navigation methods. In contrast to the predictions made in section 3.4, these KLM predictions are more specific to this particular task. The task includes five individual element selection tasks (as predicted in section 3.4), a data entry within a text element (which is the same for all techniques) and two value selections within drop-down menus. A breakdown of the task is given for each technique.

5.3.1 Using the Mouse

The task breaks down into

1. $T_H\ T_M\ T_P\ T_K$ - Selecting the type of conversion
2. $T_M\ T_P\ T_K$ - Selecting the value text box
3. $T_H\ T_M\ n_v T_K$ - Entering the value (n_v characters long)
4. $T_H\ T_M\ T_P\ T_K\ T_M\ T_P\ T_K$ - Selecting a value from the “from” drop-down
5. $T_M\ T_P\ T_K\ T_M\ T_P\ T_K$ - Selecting a value from the “to” drop-down
6. $T_M\ T_P\ T_K$ - Hitting the “convert” button

giving the formula

$$T_{Execute} = 3T_H + 8T_M + 7T_P + 7T_K + n_v T_K \quad (7)$$

where

- n_v is the number of keystrokes needed to enter the value. Based on the data used in Experiment 2 (Chapter 6), n_v is 3-letters long on average and it is the same for all techniques.
- T_H is the homing time and equals 0.4 s [6].
- T_M is the mental preparation time and equals 1.35 s for all devices [6].
- T_P is the pointing time to select a target with the mouse. It ranges from 0.8 to 1.5 s and is 1.1 s on average [6].
- T_K is the time to press a key including the mouse button. Based on Card et al. [6], T_K ranges from 0.08 to 1.20 s; however, since this range was estimated from participants' typing speed rather than a mouse click, it makes more sense to use the lower value of 0.08 s for mouse clicks. Note that the full range still applies to the set of keystrokes used for entering the conversion value (i.e., $n_v T_K$).

Given this, the estimated execution time to perform the conversion task using the mouse is between 18.4 and 26.66 s.

5.3.2 Using the Keyboard: Tab Chains

The task breaks down into

1. $T_M\ 2T_K$ - Selecting the type of conversion
2. $T_M\ 4T_K\ T_M\ T_K$ - Selecting the value text box[3]
3. $T_M\ n_v T_K$ - Entering the value (n_v characters long)
4. $T_M\ 3T_K\ T_M\ T_K\ T_M\ n_d T_K$ - Selecting a value from the "from" drop-down
5. $T_M\ T_K\ T_M\ n_d T_K$ - Selecting a value from the "to" drop-down
6. $T_M\ 3T_K$[4] $T_M\ T_K$ - Hitting the "convert" button

giving the formula

$$T_{Execute} = 11T_M + 16T_K + n_v T_K + n_d T_K \quad (8)$$

where

- n_d is the number of keystrokes required to select a value from a drop-down menu using the keyboard. n_d is the same for both keyboard navigation techniques and the actual number of keystrokes depends on how the value is selected:
    - If the arrow keys are used to select the value, the number of keystrokes depends on the size of the list (d). Since the values can be selected backwards

[3] The number of TAB strokes includes those needed to pass through the hyperlinks (see Figure 15b).

[4] The number of keystrokes (TABS) required to move from the last drop-down menu to the button that performs the conversion task depends on the selected drop-down menu. It ranges from 1 to 5 and is 3 on average.

(e.g., the last element on the list can be selected with a single backward keystroke without having to go through the who list), there are a maximum of "d / 2" keystrokes and the average number of keystrokes equals "d / 4". In this particular conversion task, there are 6-8 items on each list, thus on average 2 keystrokes will suffice to select a desired value.

- Alphabetical characters can be used to select the value by entering the first letter of a value. In this case, a few keystrokes suffice to select the desired value. More precisely, given the web page used in the conversion, only a maximum of 3 keystrokes is needed to select the desired value. Given this particular conversion task, the average number of keystrokes equals 2, which is the same as when the arrow keys are used. Note that this method is more useful for larger drop-down menus as it can significantly speed up the selection.

- n_v is the number of keystrokes required to type the value and is similar to the mouse.
- T_M is the mental preparation time, as explained above. Based on heuristics for placing M operations [6], when TAB is used more than once, T_M is calculated once before the first TAB and once before the final keystroke required to select the desired element.
- T_K ranges from 0.08 to 1.20 s. Arguably, since all strokes are on the same key when simply hitting TAB, except for the keystrokes required to enter the value (i.e., mT_K) the

lower values of T_K may be more accurate and are used to predict the execution times below.

Given the above model components, the estimated time to perform a conversion task using TAB-key navigation ranges from 16.53 to 19.89 s.

5.3.3 Using the Keyboard: ID Navigation (All Variations)

The task breaks down into

1. $T_M\, 2T_K$ - Selecting the type of conversion
2. $T_M\, 3T_K$ - Selecting the value text box
3. $T_M\, n_v T_K$ - Entering the value (n_v characters long)
4. $T_M\, 3T_K\, T_M\, n_d T_K$ - Selecting a value from the "from" drop down
5. $T_M\, T_K\, T_M\, n_d T_K$ - Selecting a value from the "to" drop down
6. $T_M\, 2T_K$ - Hitting the "convert" button

giving the formula

$$T_{Execute} = 8T_M + 11T_K + n_v T_K + n_d T_K \quad (9)$$

where

- n_d is number of keystrokes needed to select a value from a drop-down menu and is calculated similar to TAB-chain navigation.

- n_v is the number of keystrokes needed to enter the conversion value, similar to the other techniques.
- T_M is the mental preparation time to prepare to enter the ID of the element, as described in section 3.4.3.
- T_K is the keystroke time, as explained for TAB-chain navigation.

The estimated time to perform the conversion task using ID navigation is between 12.08 to 30.0 s when selection is done by releasing the activation key.

The timeout option is not estimated in this chapter as it was deemed error-prune in Experiment 1. The number of keystrokes required to enter the labels is based on the two variations of ID navigation explained in this thesis (numeric vs. 4-finger). In this particular conversion task, the only difference between these two variations is that numeric ID navigation uses one less keystroke to select the value field. This difference is negligible compared to the total execution time and was ignored.

5.3.4 Comparing the Estimated Execution Times

Table 3 summarizes the model predictions for each navigation method. With addition of a single text element, the minimum predicted result for both keyboard navigation methods is smaller than the minimum for the mouse. Based on these predictions alone, when dealing with text elements, TAB navigation is the best method overall. However, ID navigation has the lowest minimum execution time. Since the difference within the range only depends on the typing speed, for expert keyboard users, ID navigation could

potentially be the fastest method. It is also important to be reminded that ID navigation is meant to complement TAB navigation and when using ID navigation, users are always free to hit TAB when it feels faster.

Table 3: Summary of Model Predictions (s) by Method

Method	Model Prediction (s)	
	Min	Max
Mouse	18.4	26.66
Tab	16.53	19.89
ID Navigation	12.08	30.0

Chapter 6
Experiment 2: A Detailed Evaluation

6.1 Evaluation Method

A more extensive experiment was designed to compare two variations of ID navigation. One variation was an improved version of the numeric method based on results of the previous experiment (Experiment 1). The other variation was the 4-finger method explained in section 3.3. These two variations of ID navigation were compared with the existing keyboard method (TAB) as well as the mouse as a baseline condition.

To encompass the range of interactions usually found on web pages, two tasks were used. The first task was to measure the time to select a random target button from a set of buttons on a web page, as before. The second task required the participants to enter text into form elements. As with Experiment 1, both tasks were limited to a fixed view port. This experiment also included random web surfing to practice ID navigation techniques as well as a questionnaire in the end.

6.1.1 Participants

Sixteen paid participants (10 male, 6 female) performed the experiment. They were in their 20s (except one who was in early thirties) and all were relatively experienced computer users capable of using a mouse and a keyboard comfortably. Two participants were left-handed but used the mouse with their right hand.

Thirteen participants indicated they were laptop users. Of these, 3 participants indicated they used an external mouse instead of the built-in pointing device. No participant had previous experience with any technique similar to ID navigation. One participant had used the "Find As You Type" [18] feature in *Firefox* which selects hyperlinks by typing their text on the keyboard.

6.1.2 Apparatus

As with Experiment 1, this experiment was performed on a Linux-based machine in the local computer lab and Mozilla *Firefox* was used because of good keyboard support as well as expandability using extensions.

The new implementation of ID navigation with changes from section 5.2 was used in this experiment. Participants were given the option of changing the default activation (ALT) or delete key (Z). Except one participant who chose the back-quote (`) key to delete, no other participant changed the defaults. During this experiment the activation key was held to show the IDs and the characters were entered while the activation was being held. The highlighted element matching the characters entered, was selected as soon as the activation key was released.

The four input techniques were tested on two tasks: target selection ("Task 1") and data entry ("Task 2"). This was done using interactive web pages written in Javascript and stored on the local computer to avoid communication delays.

The web page used in Task 1 (target selection) consisted of a START button, a message box for instructions, and 25 target buttons. To minimize the visual scan time for all input techniques, the buttons were labeled with 25 words corresponding to 25 letters of the alphabet (A to Y) in alphabetical order. This web page was similar to the one used in Experiment 1 except the key labels did not directly correspond to the ID labels.

Data collection was significantly improved during Experiment 2. For example, instead of measuring the selection time from the moment the start button was clicked, it was measured from the first use of the input technique after the start button was clicked. Or for example, instead of simply recording incorrect button presses as the only source of error rate, mouse clicks in the space between the buttons were also recorded.

Figure 15a shows the experiment web page when the activation key is held down and the numeric IDs are shown.

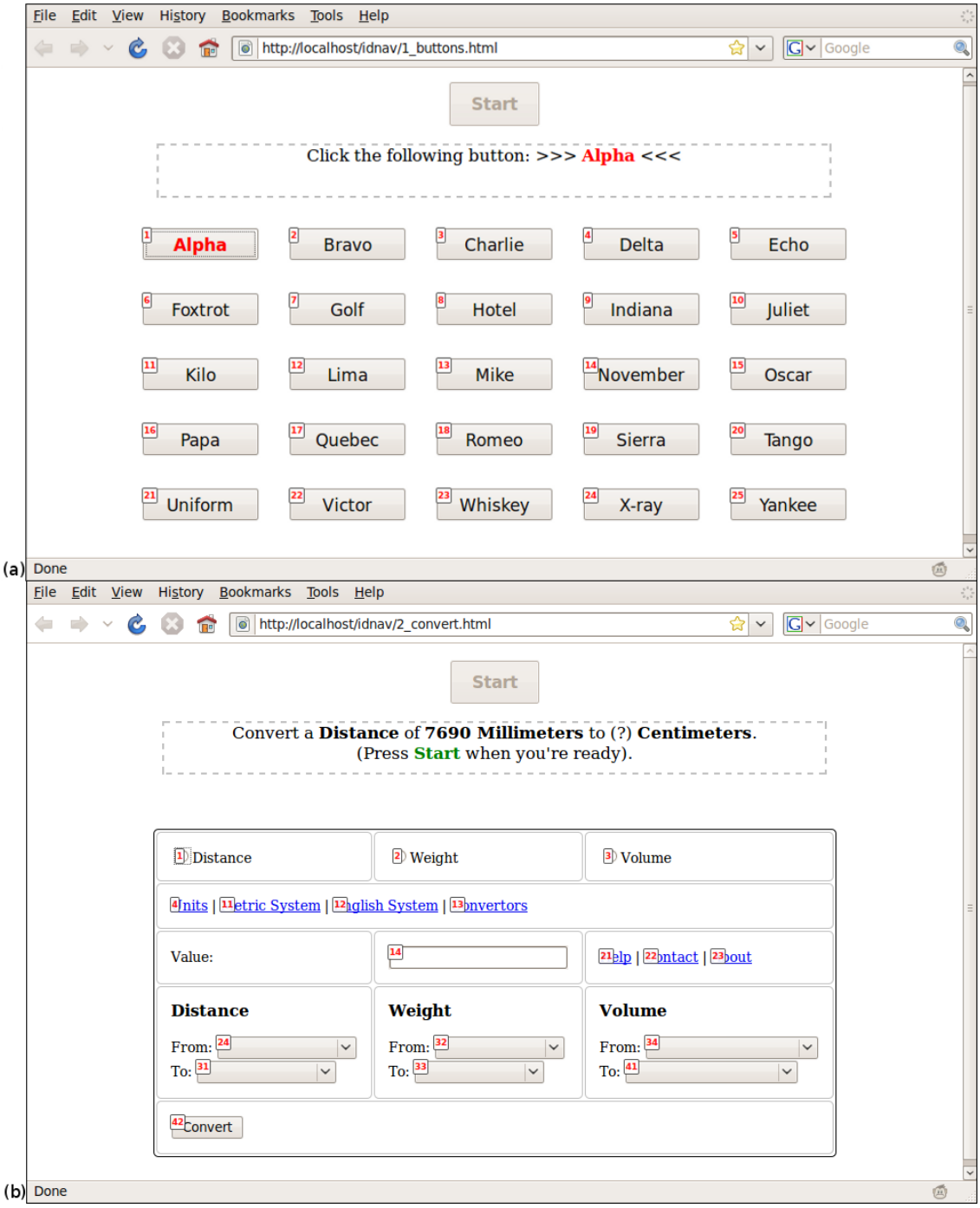

Figure 15: (a) The experimental interface during Task 1 (using numeric Ids). The activation key is down causing the IDs to appear beside each focusable web page element.(b) The interface during Task 2 (using the 4-finger technique). (Experiment 2)

Initially, only the START button was enabled on the page. Once participants clicked the START button, it was disabled and the target buttons were enabled with the button participants were expected to press shown in red and bold. Once a target button was selected, the message box indicated if the correct button was pressed. The participants were then instructed to start the next trial when ready.

The target button, the button actually pressed, and the selection time were logged for each trial. For all keyboard techniques, the selection time started from the first key event after the START button was pushed. When the mouse was used, selection time started when the pointer moved beyond a certain threshold (in any direction) after the START button was clicked. This method of measuring selection time, does not measure the initial mental preparation time that was used in the predictions.

In Task 2 (data entry), participants were presented with a form to convert a value from one unit to another. The task was to select the type of conversion (distance, weight or volume), enter the value, select the units from a corresponding drop-down menu and press a convert button. A number of random hyperlinks were added in two parts of the form to require participants to "jump" to a different section. Figure 15b shows the web page when 4-finger ID navigation is used with the IDs shown (i.e., the activation key is down).

6.1.3 Procedure

The experiment was performed in two one-hour sessions per participant. To avoid confusion due to the similarity of the ID navigation techniques, each session included one ID navigation technique (numeric or 4-finger) and either the mouse or TAB technique. The sessions were performed with a break of at least two hours but no more than two days. This was considered long enough to avoid fatigue and confusion between ID navigation techniques, yet short enough for participants to remember the first session when filling out the questionnaire.

At the beginning of each session the appropriate ID navigation technique was explained to participants and they were asked to practice the technique on random web sites. They were asked to browse different sites and try the technique on various pages and page elements including different types of form elements. A couple of participants who weren't familiar with TAB navigation were asked to practice that method as well. Once participants felt comfortable with the techniques, they performed a few test trials of the experiment using both techniques for the current session. Both tasks were completed for one technique, then after a break of a few minutes the other technique was tested.

Participants were instructed to press the START button and proceed as quickly and accurately as possible. They were told that at the end of each trial, they could stop and relax before pressing START again. If an error was made, such as an incorrect button selection or a mistake in the conversion input, it was indicated in the message box but the

participants continued without redoing the trial. At the end of each block of trials, participants were asked to take a short break and continue when ready. To force breaks and avoid fatigue, the START button remained disabled for five seconds after each trial block ended.

To keep TABs within the web page, initially the START button was focused. After START was pushed and disabled, the first target button (or form element) was focused. As this is slightly different from normal TAB behaviour of the browser (where the address bar is focused), it was explained to the participants so they would not have to visually scan for the focus indicator.

6.1.4 Design

Since the interactions are substantially different between Task 1 and Task 2, they were treated as two separate experiments.

Task 1: Target selection

Task 1 was a 4 × 6 × 12 within-subjects design. The factors and levels were as follows:

Input Technique	ID navigation (numeric), ID navigation (4-finger), TAB, Mouse
Block	1, 2, 3, 4, 5, 6
Trial	12 buttons (all odd/even buttons in alternating blocks) in random order

The dependent variables for Task 1 were time to select a target button (selection time) and the mean error rate (%). The total number of trials was 4608 (16 participants × 4 input techniques × 6 blocks × 12 trials).

To minimize skill transfer, the input techniques were assigned to participants using a Latin square such that each session included only one ID navigation technique.

Task 2: Data entry

Task 2 was a 4 × 6 × 4 within-subjects design. The factors and levels are as follows:

Input Technique	ID navigation (numeric), ID navigation (4-finger), TAB, Mouse
Block	1, 2, 3, 4, 5, 6
Trial	4 conversion tasks (presented in random order)

The dependent variable for Task 2 was "task completion time": the time to perform a single conversion task starting when the first form element (conversion type) was accessed. The total number of trials was 1536 (16 participants × 4 input techniques × 6 blocks × 4 trials). While error rate was originally measured during the experiment, no error relating to a particular input technique was recorded. Thus, the error rate was not analysed further.

6.2 Results and Discussion

In Task 1 (target selection), the grand means for the dependent variables were 1.37 s for selection time and 0.85% for error rate. In Task 2 (data entry), the grand mean for task completion time was 12.2 s.

6.2.1 Selection Time (Task 1)

As with Experiment 1, TAB navigation was the slowest technique with a mean selection time of 2.2 s. The selection time was 40% less for 4-finger ID navigation (1.31 s), 49% less for numeric ID navigation (1.12 s) and 60% less for the mouse (0.87 s). The differences were statistically significant ($F_{3,45} = 95.8$, $p < .00001$). See Figure 16.

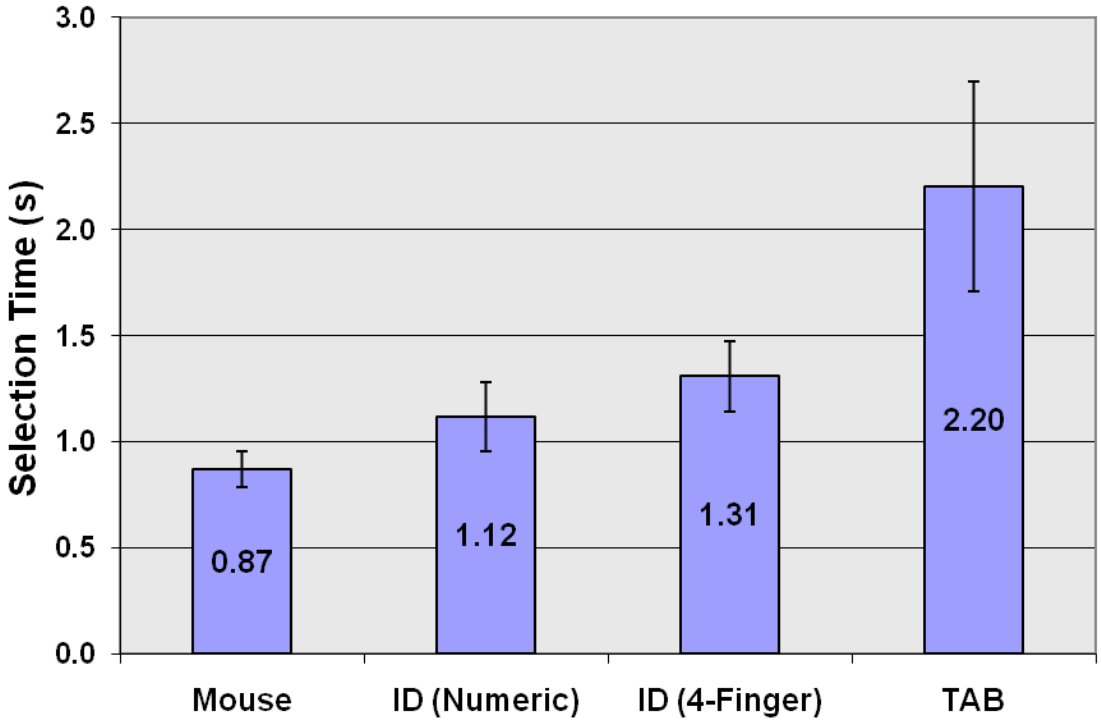

Figure 16: Mean selection time by input technique (Experiment 2)
(Note: error bars show ±1 standard deviation)

Based on selection time in this experiment, ID navigation is clearly a good alternative to TAB navigation. The mouse was only 22% faster than numeric ID navigation, while both techniques were at least twice as fast as TAB navigation. Similar to Experiment 1, the number of elements in the test web page was about half the estimated average for a normal web page [2,16] and larger difference are still expected for real web pages.

In comparison to Experiment 1, the values measured for selection time are smaller in this experiment. This is because in this experiment selection time is measured from the moment the interaction technique is used. The initial mental preparation time is not measured in this experiment, while it was measured in Experiment 1. The relative

difference between the selections times is very similar to Experiment 1. In fact, the difference between (numeric) ID navigation and TAB navigation was only 2% less than Experiment 1.

There was a significant effect of block on selection time for all techniques in this experiment ($F_{5,75} = 20.8$, $p < .00001$). As shown in Figure 17, the slope of the lines is relatively parallel. This means participants' speed improved with practice and the improvement was the same regardless of the technique used.

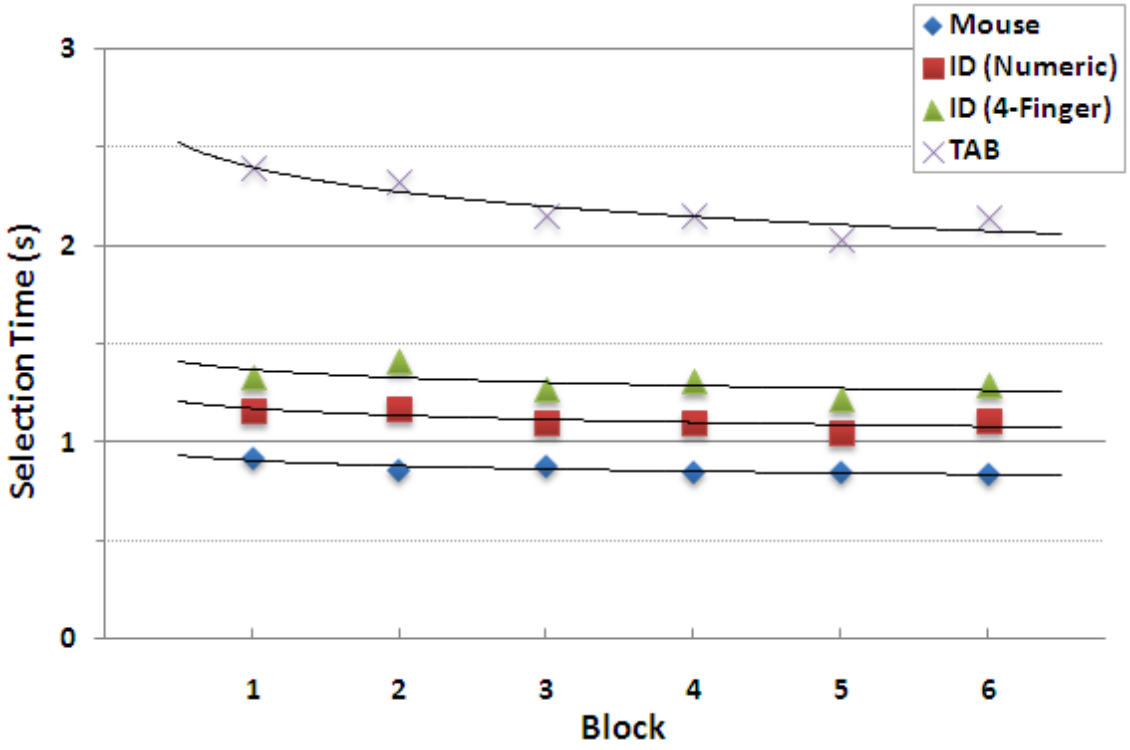

Figure 17: Selection time by block (Experiment 2)

Numeric ID navigation was 14.5% faster than 4-finger ID navigation. Not everyone who participated in the experiment was comfortable typing with all four fingers and some

could only use their index and middle fingers. Thus touch typists are expected to perform faster with 4-finger ID navigation.

Order of the buttons

The order of the buttons (their TAB order on the page) is expected to effect the selection time for TAB navigation more significantly than for other methods. However, TAB order is not the same as the pointing distance for the mouse condition. For example, button "Charlie" in Figure 15a was closest to the START button, while "Alpha" had the smallest TAB order. To reveal these distinctions, we can consider the effect of button "row" on selection time. The average TAB order of the buttons in each row is relative to their average pointing distance from the START button.

Figure 18 shows the relationship between selection time and row. While the effect of row on selection time was significant for all techniques ($F_{4,60} = 113.2$, $p < .00001$), the linear relationship for TAB chains is clearly much steeper than for all other techniques. The slope of the regression line for TAB (0.55) was 89% steeper than for both the mouse (0.06) and numeric ID navigation (0.06). 4-finger ID navigation was also significantly less steep than with TAB navigation but slightly steeper than with the other two techniques (0.14). This is because the number of digits in the IDs grows faster for this technique. This requires more keystrokes for buttons farther down the page.

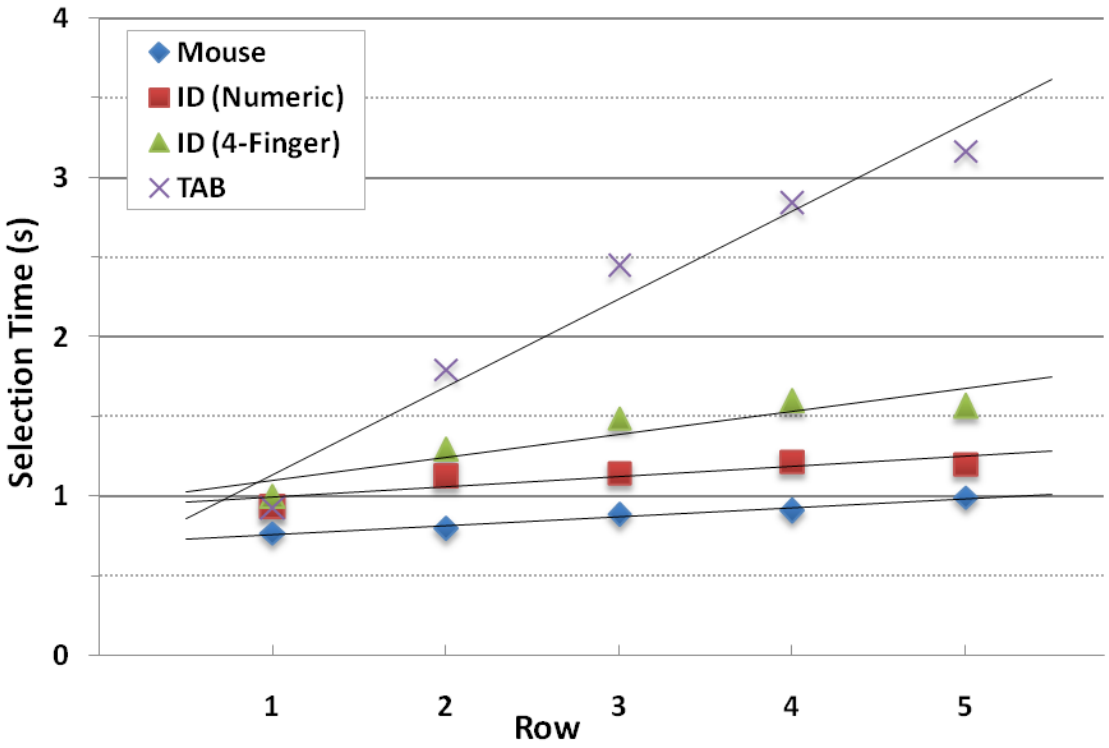

Figure 18: Selection time by row and input technique (Experiment 2)

In Experiment 1, an interesting strategy of pressing and holding the TAB key until the target was almost reached was observed. To allow participants to use the techniques as they normally would, this was allowed during this experiment. Interestingly, it did not seem to help as in many cases participants overshot the target and had to backtrack using SHIFT. This reverse traversal was unnatural and seemed to take much longer. In this case, some participants even had to visually scan for the SHIFT key.

Comparing the observations with the predictions

In this experiment, the observations for both traditional methods fall close to the lower end of the predictions from the keystroke-level model (Table 4). The mean selection time

for the mouse was only 1% less than predicted. Optimal performance for the mouse was expected as the buttons were relatively large. This reduces the actual pointing time according to Fitts' Law [20], but is not accounted for in the pointing operator in Card et al.'s [6] original model. The empirical result for TAB navigation was 10% less than the minimum predicted selection time for a page with 25 elements. This is likely because the predictions assume one TAB press at a time, while some participants chose to hold TAB. Additionally, since the predictions are based on typing speed of the participants, repeatedly pushing a single button should be faster. Thus, for users with disabilities or poor keyboarding skills, the selection time using TAB navigation might be even longer.

Table 4: Model Predictions vs. Observations (Experiment 2)

Method	Model Prediction (s)		Observation (s)
	Min	Max	
Mouse	0.88	1.58	0.87
Tab			
25 elements	2.47	18.15	2.2
ID Navigation			
(2-digit IDs + activation key)	0.24	3.6	1.12 / 1.34

Note: The initial mental preparation time is not included in this table since it was not measured in Experiment 2.

For both ID navigation techniques, the observations fall within the range of the predictions and below the average. The mean selection time for numeric ID navigation

was 33% less than the average and the mean selection time for 4-finger ID navigation was 22% less.

These predictions did not take into account the implementation-specific response time needed to load the IDs. While the new implementation was much faster than the original extension used in Experiment 1, both implementations were developed as extensions to the web page itself using JavaScript. While in neither experiment participants noticed a delay with the system, with a faster low-level implementation of ID navigation, a shorter selection time can be expected in the measurements.

The model assumes expert users and does not take into account "user unpredictability" due to factors such as fatigue and other surrounding factors. While this was a controlled experiment, fatigue could not be avoided or quantitatively measured. In an initial pilot of this experiment, the performance decreased towards the end of the experiment due to fatigue. To minimize fatigue in the actual experiment, more blocks were used with less trials per block.

Overall, the model provided very good predictions of the *relative* performance of all input techniques. The linear relationship between selection time using TAB and the position of buttons is also clearly visible in both experiments.

6.2.2 Error Rate (Task 1)

With invalid button selections as the only errors, mouse navigation was the most accurate technique with no errors. The next most accurate technique was numeric ID navigation (0.4%) followed by TAB navigation (1.0%). 4-finger ID navigation was the least accurate method with an error rate of 1.9%. See Figure 19. The differences were statistically significant ($F_{3,45} = 6.61$, $p < .00001$).

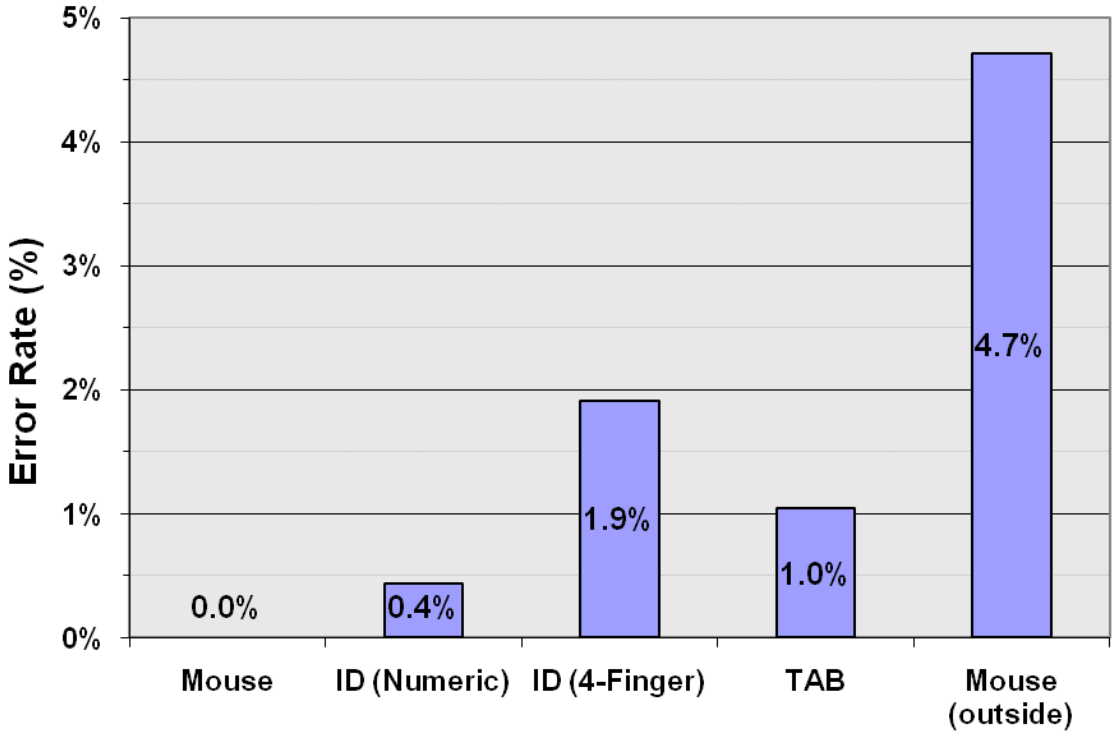

Figure 19: Error rate by input technique (Experiment 2)

In this experiment not only the error rate was much lower than Experiment 1, errors were random and did not follow a particular pattern. The small percentage of errors for numeric ID navigation were simply due to accidental typing errors when entering

numbers. The larger error rate for 4-finger ID navigation was due to typing errors as well, except in this case participants were more likely to make typing mistakes. This was especially the case for participants who could not use their four fingers to type. During TAB navigation hitting TAB on the wrong element, normally the element just before or after the target element was the main source of errors. Most errors for this technique occurred when TAB was held down instead of being pressed multiple times. This shows while holding TAB can slightly speed up selection of elements that require a longer TAB-chain, it has disadvantages when it comes to accuracy.

Clicking the mouse in the space between buttons occurred with reasonable frequency, thus the error rate for the mouse above is artificially low. On most real web pages, elements are typically smaller and closer together, so the true error rate for the mouse would likely be higher. On the other hand, having small, tightly packed elements should not affect keyboard navigation techniques. For the purpose of comparison, a new error rate was calculated for the mouse by counting clicks between buttons. In this case the error rate was 4.7% and significantly higher than all keyboard navigation techniques (the right-most column in Figure 19).

In contrast to Experiment 1 (10% error rate for ID navigation), the new implementation of (numeric) ID navigation was 96% more accurate. This was achieved by releasing the activation key instead of using a timeout to confirm selections, as well as other

improvements in the implementation. While a method was implemented to correct selection mistakes, participants rarely made corrections.

6.2.3 Data Entry Time (Task 2)

Even in Task 2 (data entry), the mouse was the fastest method with a mean task completion time of 9.8 s. The task was completed in 11.7 seconds using numeric ID navigation, 12.1 seconds using TAB-key navigation, and 15.0 seconds using 4-finger ID navigation. The differences were statistically significant ($F_{3,45} = 45.5$, $p < .00001$). See Figure 20.

A Scheffé post hoc comparison test was performed to determine which pairs of conditions were significantly different. Significant differences were found between all pairs except numeric ID Navigation and TAB-key navigation ($p > .05$).

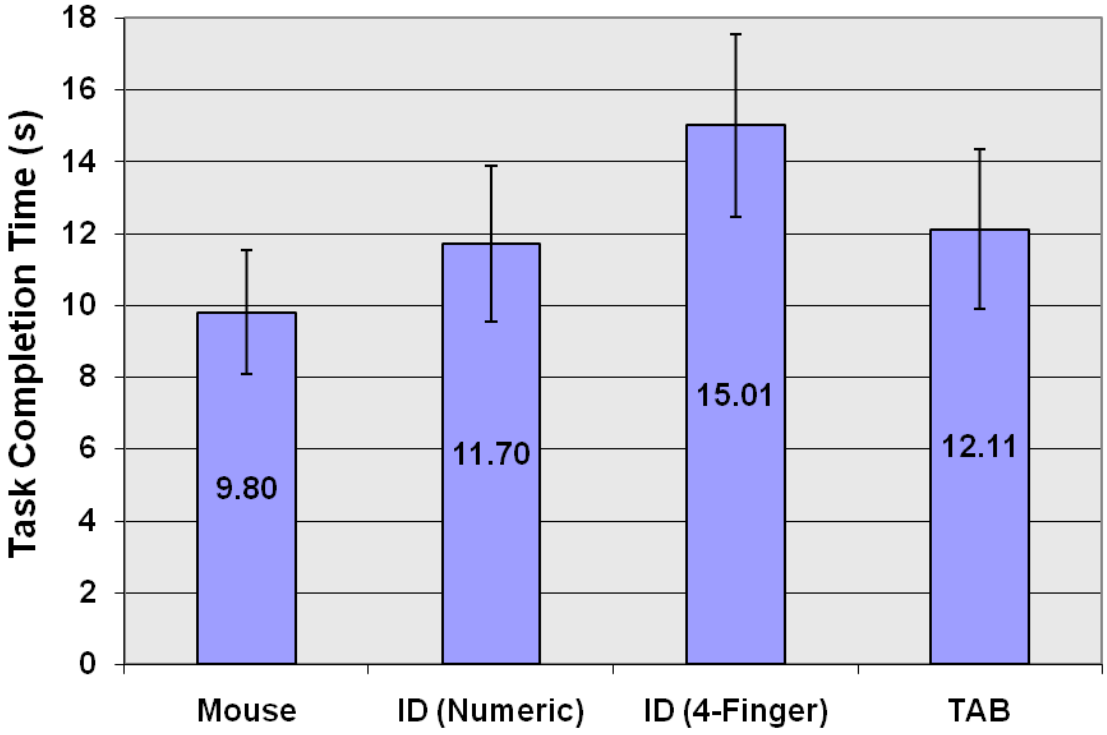

Figure 20: Mean Task 2 completion time by input technique (Experiment 2)
(Note: error bars show ±1 standard deviation)

Keyboard methods are favoured with an increasing number of text entry elements. With the addition of a single text entry field in Task 2, the relative difference between the mouse and numeric ID navigation (16%) was less than the difference during Task 1 (22%).

While numeric ID navigation was 74% faster than TAB in Task 1 (target selection), the difference between the two in Task 2 was not significant, as just noted. This was expected, as in a web form with many data entry elements accessed in sequence, TAB is the ideal method to traverse the elements. That is why ID navigation is meant to improve TAB navigation, not replace it.

The slow task completion time for 4-finger ID navigation was mainly due to the numeric nature of Task 2. Most participants confused actual number keys (1 to 4) with the labels used for ID navigation. Some even wanted to use the first four number keys to navigate.

There was a significant effect of block on task completion time for all four techniques ($F_{5,75} = 26.1$, $p < .00001$). See Figure 21. The regression lines are relatively parallel, meaning the learning effect was the same regardless of the technique used.

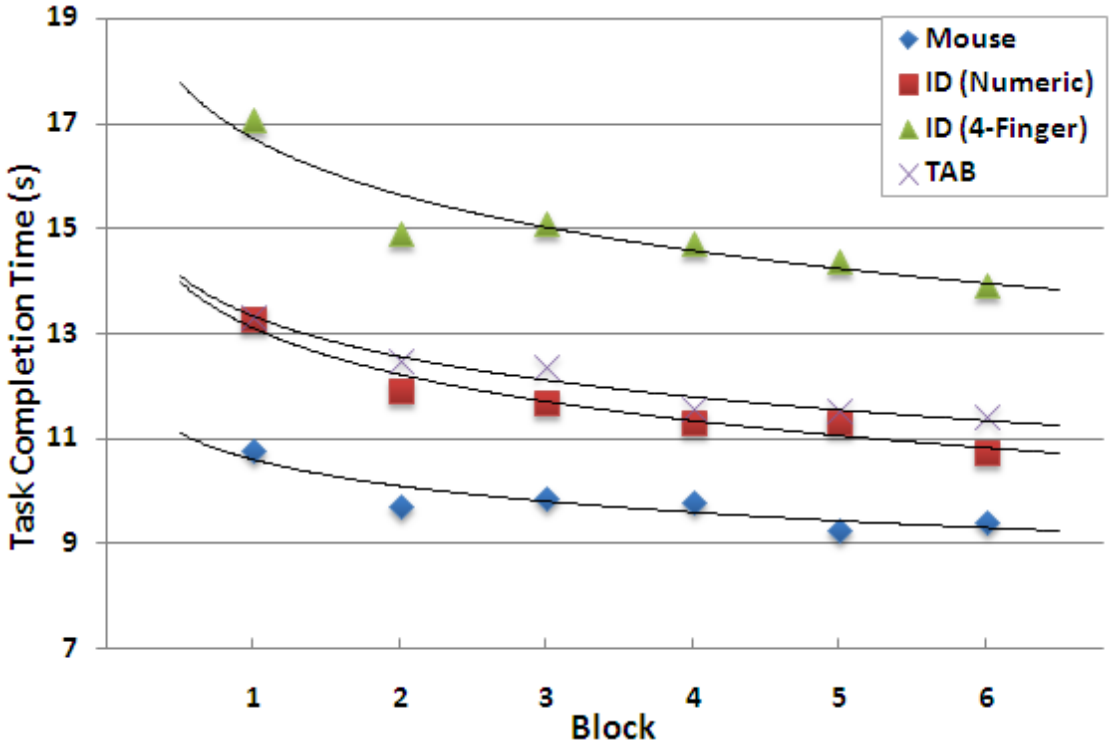

Figure 21: Task 2 completion time by block (Experiment 2)

Comparing the Observations with the Predictions

Using mouse and TAB navigation, the observed task completions times were much lower than those predicted in the previous chapter (5.3). For numeric ID navigation the

observed task completion time was lower but closer to the lowest predicted value. The task completion time measured for 4-finger ID navigation fell within the range of the predictions. However, since other factors have likely caused this method to be slower than the other techniques, all observations could be considered faster than the predicted results. While the predictions for target selection task (Task 1) were slower than the observations as well, the difference was much larger in the unit conversion task.

One of the most significant factors in these predictions was the mental preparation time (1.35 s). When predicting target selection only a single mental preparation variable was present, while when predicting the conversion task 8-11 mental preparation variables were present. It is possible that the large difference between the observations and the predictions for the unit conversion task was caused by an estimated value of mental preparation time that was larger than the actual mental preparation time.

Use of a numeric keypad

During the experiment, participants were instructed to use their preferred numeric entry method. All participants used the numeric keypad to enter numbers in the form and all but one used it for ID Navigation as well. Based on these results, *exclusive* use of the numeric keypad for ID navigation is not recommended, since it is already the preferred choice of numeric entry for many users. This is consistent with Experiment 1.

When used with an activation key however, the numeric keypad was a fast choice despite the small homing time necessary to switch between the main keyboard and the numeric keypad.

6.2.4 Questionnaire

In a questionnaire, participants were asked to rank the four techniques based on their personal preference. Overall, 14 participants preferred the mouse as their first choice and 15 chose ID navigation as their second choice (or better). TAB and 4-finger ID navigation took the 3rd and 4th spots with more participants in favour of TAB. See Table 5.

Table 5: Participants' overall ranking of all four methods (Experiment 2)

Method	Rank			
	1	2	3	4
Mouse	**14**	1	1	0
ID Navigation (numbers)	2	**13**	1	0
ID Navigation (4-finger)	1	0	**6**	**9**
Tab	1	1	**9**	**5**

These results contradict the preference of ID navigation over the mouse in Experiment 1. This was possibly due to the larger number of tasks in Experiment 2. In both experiments ID navigation was new to the participants. However, those who participated in Experiment 1 ranked a new technique that was showcased to them, while those who

participated in Experiment 2 ranked the new method right after they had extensively used it in a single session. Another reason might be the similarity of the two ID navigation techniques causing the bad experience of some users with 4-finger ID navigation to effect the ranking of numeric ID navigation as well.

In a similar comparison, participants were asked to rank all methods only when constant data entry was needed. The results were similar but less unanimous. While 11 participants ranked the mouse as their first choice, 7 preferred a keyboard method. See Table 6.

Table 6: Participants' ranking of methods for tasks requiring data entry (Experiment 2)

Method	Rank			
	1	2	3	4
Mouse	**11**	4	1	0
ID Navigation (numbers)	3	**8**	4	1
ID Navigation (4-finger)	0	1	**6**	**9**
Tab	4	2	**6**	4

On a 7-point Likert scale (1 = least comfortable, 7 = most comfortable), participants ranked their level of comfort with all four methods. The mouse was the most comfortable method with an average rating of 6.7, followed by numeric ID navigation (5.4) and TAB (4.8). 4-finger ID navigation was the least comfortable method (3.4). See Table 7. The

perceived level of comfort for numeric ID navigation in Experiment 2 was exactly the same as the participant response from Experiment 1 (5.4 out of 7).

Table 7: Participants' perceived level of comfort with all methods (Experiment 2)

(1 = least comfortable, 7 = most comfortable)

Method	# Responses per Rank							Mean
	1	2	3	4	5	6	7	Rank
Mouse			1			1	14	**6.7**
ID Nav'n (numbers)		1		2	4	7	2	**5.4**
ID Nav'n (4-finger)		3	6	4	3			**3.4**
Tab	1	1	2	2	3	4	3	**4.8**

The participants were asked if they would enable ID navigation on their own browser. Of the participants who answered yes, 8 preferred numeric IDs while 2 preferred custom IDs. They were also asked to suggest alternate IDs. Most wanted to keep numeric IDs, some suggested the left-hand version of the 4-finger method (A, S, D, F) and some suggested 4-finger combinations on the numeric keypad such as (7, 8, 9, +) and (NUM_LOCK, /, *, -).

Participants were asked to compare the highlighted selection indicator to *Firefox*'s default dashed border. Nine participants preferred the improved method, however only two felt the improved feedback was “required”.

Finally, they were asked for suggestions to improve ID navigation. One participant thought the large number of digits in the labels (4-finger method) was confusing. On the other hand, a number of participants preferred a numeric method with a smaller subset of digits (e.g., 1-4).

Chapter 7
Conclusion and Future Work

As presented in both experiments, ID navigation can reduce the selection time, compared to TAB navigation, to a constant time. ID navigation can improve existing keyboard navigation techniques to select an arbitrary element on screen without the need for a complex syntax. It can even complement pointing devices for expert keyboard users.

Numeric ID navigation was found far superior in performance to the 4-finger method, especially during a task already involving numeric entry. However, the 4-finger method should not be abandoned as it may be the only choice in certain applications such as chord keyboards that can benefit disabled users. Additionally, with enough practice, the 4-finger method may be more comfortable for the touch typist and the left-handed variation of 4-finger ID navigation can help them reduce the load on the right hand.

In the future, it may be helpful to compare alternate choices of labels as well as special input devices for the 4-finger method. It may also be worthwhile to compare ID navigation with slower pointing devices used in notebook computers (e.g., touchpads and pointing sticks). ID navigation can be easily extended to use speech recognition. It would be interesting to see an empirical evaluation of numeric ID navigation using speech recognition, both using spoken digits and natural numbers.

Bibliography

1. US department of justice. Section 508 of the federal rehabilitation act. http://www.section508.gov/.

2. Google code: Web authoring statistics. http://code.google.com/webstats/.

3. Amadio, P. C., Frymoyer, J., Szabo, R. M., and King, K. J. *Repetitive stress injury*. The Journal of Bone and Joint Surgery, 2001.

4. Andersen, J. H., Thomsen, J. F., Overgaard, E., et al. Computer Use and Carpal Tunnel Syndrome: A 1-Year Follow-up Study. *The Journal of the American Medical Association, 289*, 22 (2003), 2963-2969.

5. Bilmes, J. A., Li, X., Malkin, J., et al. The Vocal Joystick: A voice-based human-computer interface for individuals with motor impairments. *Human Language Technology Conf. and Conf. on Empirical Methods in Natural Language Processing*, (2005).

6. Card, S. K., Moran, T. P., and Newell, A. The keystroke-level model for user performance time with interactive systems. *Communications of the ACM, 23*, 7 (1980), 396-410.

7. Card, S. K., Moran, T. P., and Newell, A. *The psychology of human-computer interaction*. Hillsdale, NJ: Erlbaum, 1983.

8. Chisholm, W., Vanderheiden, G., and Jacobs, I. Web content accessibility guidelines 1.0. *interactions, 8*, 4 (2001), 35-54.

9. Dix, A. J., Finlay, J., Abowd, G. D., and Beale, R. *Human-computer interaction* (2nd ed.). London: Prentice Hall, 1998.

10. Gandhi, M. and Jacob, J. Natural number recognition using MCE trained inter-word context dependent acoustic models. *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal*, (1998), 457-460 vol.1.

11. Hackett, S., Parmanto, B., and Zeng, X. Accessibility of Internet websites through time. *ACM SIGACCESS Accessibility and Computing,*, (2003), 32-39.

12. Hemphill, C. T. and Thrift, P. R. Surfing the Web by voice. *Proceedings of the third ACM international conference on Multimedia*, (San Francisco, California, United States: ACM, 1995), 215-222.

13. Huang, X., Acero, A., and Hon, H. W. *Spoken language processing*. Prentice Hall PTR Upper Saddle River, NJ, 2001.

14. Inc, C. S. M. *Java Look and Feel Design Guidelines with CD-ROM*. Addison-Wesley Longman Publishing Co., Inc., 2001.

15. Keele, S. W. and Posner, M. I. Processing of visual feedback in rapid movements. *Journal of Experimental Psychology, 77*, 1 (1968), 155-8.

16. Kelly, B. WebWatching UK universities and colleges. 1997. *Ariadne Magazine (Web Version)*. http://www.ariadne.ac.uk/issue12/web-focus/.

17. Lauke, P. H. Evaluating Web Sites for Accessibility with Firefox. *Ariadne, 44*, (2005).

18. Leventhal, A. Mozilla keyboard feature: Find as you type. http://www.mozilla.org/access/type-ahead/.

19. MacKenzie, I. S. and Guiard, Y. The two-handed desktop interface: are we there yet? *Conference on Human Factors in Computing Systems*, (ACM New York, NY, USA, 2001), 351-352.

20. MacKenzie, I. S. and Buxton, W. Extending Fitts' law to two-dimensional tasks. *Proceedings of the ACM Conference on Human Factors in Computing Systems*, (New York: ACM, 1992), 219-226.

21. Nilsson, R. and Racine, S. Use of keyboard for mouseless data entry in UI design. *Conference on Human Factors in Computing Systems*, (New York: ACM, 2006), 135-140.

22. Noe, R. Mouseless browsing extension. http://www.rudolf-noe.de/index.php?/content/view/14/26/.

23. Olsen, D. R. *Developing user interfaces*. Morgan Kaufmann, 1998.

24. Raggett, D., Le Hors, A., and Jacobs, I. HTML 4.01 specification. *W3C Recommendation REC-html401-19991224, World Wide Web Consortium (W3C), Dec,*, (1999).

25. Riviere, C. N. and Thakor, N. V. Effects of age and disability on tracking tasks with a computer mouse: Accuracy and linearity. *Journal of rehabilitation research and development, 33*, 1 (1996), 6-15.

26. Schrepp, M. and Fischer, P. A GOMS model for keyboard navigation in web Pages and web applications. *Proceedings of the 10th International Conference on Computers Helping People with Special Needs*, (Linz, Austria: Springer, 2006), 287.

27. Shneiderman, B. Universal usability. (2000).

28. Shneiderman, B. The future of interactive systems and the emergence of direct manipulation. *Human Factors and Interactive Computer Systems: Proceedings of the Nyu Symposium on User Interfaces, New York, May 26-28, 1982*, (Intellect Books, 1984).

29. Spalteholz, L., Li, K. F., and Livingston, N. Efficient navigation on the world wide web for the physically disabled. *Proceedings of the 3rd International Conference on Web Information Systems and Technologies*, (2007), 321–326.

30. Spalteholz, L., Li, K. F., Livingston, N., and Hamidi, F. Keysurf: A character controlled browser for people with physical disabilities. *Proceedings of the 17th International Conference on World Wide Web*, (New York: ACM, 2008), 31-40.

31. Treisman, A. M. and Gelade, G. A feature-integration theory of attention. *Cognitive psychology, 12*, 1 (1980), 97-136.

Appendix A:
Implementation Changelog

This appendix is a detailed list of changes made to the re-implimented version of ID navigation used in Experiment 2 (Chapter 6). While many of these changes were never used in the empirical evaluation presented in this thesis, they improve user experience with ID navigation when used with real web sites. Most of these improvements are based on small pilot studies performed during the implementation. Thus, it is important to note these improvements and apply them when implementing ID navigation or a similar technique.

- ID labels presentation and style improved.
- Autoclick elements (enabled by default).
- Disable timer by default (type elements as longs as activation is held and select when released).
- A deleted key clears the selection currently in the buffer.
- Clear last selection when TAB is pressed.
- Reset label positions after window is modified (e.g., resized or lost focus).
- Option to cache labels or recreate them everytime the activation key is down. The option is slower but is needed for more dynamic websites.

- Increased maximum number of labels. The labels can be limited to load faster on extremely large web sites. If the maximum number of labels is limited to *n*, only the first *n* elements on top of the page will have labels assigned to them.
- Added option to highlight selected element or use the default focus indicator (highlighting enabled by default).
- Transparent highlighting to keep background images (if any) visible.
- Added support for more elements such as image maps, [div/img].onclick and the "button" tag used in some dynamic web sites.
- Remove labels from elements that are disabled, hidden or otherwise useless (e.g. bookmarks that are created using "href" tags).
- Fixed label positions for "fixed" elements when the page is scrolled.
- Improved overall user interface: added preferences and help dialogs to Firefox.
- Allow labels to use arbitrary characters (instead of key/character names)
- Allow users to change the style of labels (e.g., increase size, use high-contrast colors, etc).
- Account for unexpected user behaviour (e.g., labels typed too fast or at the same time, the activation key is released before the (last) label key is released).