THE UNIVERSITY OF CALGARY

Integrating Back, History and Bookmarks in Web Browsers

by

Shaun Kaasten

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

SEPTEMBER, 2001

THE UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "Integrating Back, History and Bookmarks in Web Browsers" submitted by Shaun Kaasten in partial fulfillment of the requirements for the degree Master of Science.

[illegible]

Supervisor, Dr. Saul Greenberg
Department of Computer Science

[illegible]

Dr. Sheelagh Carpendale
Department of Computer Science

[illegible]

Dr. Jeff Caird
Department of Psychology

[illegible]

External Examiner, Dr. Andy Cockburn
University of Canterbury

Nov 21, 2001
Date

Abstract

While there has been much discourse on web site usability, there has been comparatively little done about the usability problems that plague conventional web browser software that people use to browse the web. This is troubling, as gains that can be made on the browser software will improve the usability of all web sites. Most web browsers include Back, History and Bookmark facilities that simplify how people return to previously seen pages. While useful in theory, studies have found that users do not take enough advantage of these facilities. This thesis examines the usability issues present in these revisitation facilities. These issues include the disparate models that the user must understand to operate them and the mismatch between how they represent pages and how users remember their pages. To explore this issue, we ran an experiment where we compared how well people could recognize previously visited pages when shown its title, URL address or thumbnail image. The results of this experiment were translated into the design and implementation of our own alternative revisitation system. It is based on the single model of a recency-ordered history list to integrate Back, History and Bookmarks. Enhancements include: Back as a way to step through this list; implicit and explicit 'dog-ears' to mark pages on the list (replacing Bookmarks); searching/filtering the list through dynamic queries; and visual thumbnails to promote page recognition.

Acknowledgments

First and foremost, I would like to thank Saul Greenberg, for his expertise, insights, motivation and patience. It's been a pleasure, and I'm glad that our work together is not over yet.

I would like the thank all of the members of Grouplab for being everything you could want in colleagues and friends. In particular, Ana for bringing a much needed Brazilian flare to the lab, Chester for bringing a much needed rural Alberta flare to the lab, James for bringing endless enthusiasm and curiosity, and Mike Rounding for being the fun guy in the lab.   I especially want to thank Mike Boyle for the many hours he spent reading, coding, and discussing this research. Also, I'm thankful that Chris decided to join our lab as a psychology source, and decided to hang around as a friend.

Twenty volunteers from the department of Computer Science participated in my study. I would like to thank them for participating and contributing to this research.

Kent Sullivan and Robert Graf from Microsoft provided support and encouragement for this research. I especially would like to thank them for inviting me to visit Microsoft and speak with the Internet Explorer and Windows Shell teams.

Finally, this research is built on the work of Linda Tauscher and Andy Cockburn.   I hope this research meets the bar they have set.

Dedication

This thesis is dedicated to my parents, Tony and Bev Kaasten.

Table of Contents

List of Tables

List of Figures

1. Introduction

1.1 The situation

People regularly go back to web pages that they have previously visited. Between 58–81% of users' navigations are revisits (Tauscher and Greenberg, 1997; Cockburn and McKenzie, 2000). The design of commercial web browsers acknowledges this phenomenon. Virtually all include somewhat standard *revisitation facilities* to help users return to previously visited web pages: the Back/Forward buttons, the History list, and Bookmarks. However, all of these facilities are not on equal ground. While users rely heavily on Back, accounting for 30% of navigations, other revisitation facilities are used rarely (Tauscher and Greenberg, 1997). History and Bookmarks account for a meagre 1% and 2% respectively of user navigations. This indicates that users are not adopting these facilities and are instead using cruder ways to revisit pages such as retracing their steps through hyperlinks.

1.2 The problems

While the current revisitation facilities are useful, we believe they have several flaws that limit their usability.

1. **There is a lack of integration between revisitation facilities**. The Back button, History and Bookmarks all use dissimilar underlying models and interfaces. While it is common (and often beneficial) for a system to have several ways to accomplish a task, users can find it hard to make sense of facilities that follow different rules.
2. **There are identified problems with the session-based stack model of the Back/Forward buttons**. Previous research has identified problems with the stack model that underlies the Back/Forward Buttons. (Cockburn and Jones, 1996). First, the stack model may pop previously visited pages off the stack, which means they are no longer reachable. Second, the session-based model means that pages seen in previous browsing sessions are not accessible via Back. Third, most people have a naïve mental model of Back, thinking it works as a recency-based list rather than a stack. This

sometimes leads to confusion about what pages can or cannot be accessed via the Back button.

3. **Recognizing particular pages in the history list is difficult**. History lists automatically collect pages as a person visits them and present them within some kind of list. Yet, identifying the desired page can be difficult for several reasons (Cockburn and Greenberg 1999). The main problem (somewhat shared by bookmarks) is that a person must visually scan a long list to find a desired page. People may have difficulty recognizing the page representation (e.g., its URL or its title) as it may not match how a person remembers the page.

4. **Finding particular pages in the history list is difficult**. History systems order pages a variety of ways: alphabetically by title or URL, by recency, by number of visits, or by web domain hierarchy. The order of pages may position the page so that it is difficult to find— i.e., far down the list, or embedded deep within a hierarchy.

5. **Bookmarks have several recognized problems**. First, to become a bookmark, people must mark a page explicitly as a bookmark as it appears in the browser window. This is a heavyweight explicit action requiring the user to immediately decide that a particular page has future importance. If a person does not mark the page (and the majority are not) he or she must reacquire it through some other mechanism. Second, bookmark lists require constant maintenance from the user—e.g., to prune unwanted pages, appropriately rename them, or sensibly organize them. This extra work is something that many people are unwilling to do (Abrams, Baecker and Chignell 1997). Third, similar to History, desired bookmarks may be difficult to find.

6. **Many alternatives to the standard revisiting facilities require heavy use of screen real estate**. While the current generation of utilities are somewhat space-conservative, a variety of novel systems use space-intensive visual representations of web history and are unlikely to be used in practice (Cockburn and Greenberg 1999). Although screens sizes, resolution and multi-monitor support are increasing, most users are reluctant to devote screen real estate to these space-greedy revisiting utilities.

1.3 Goals

I will address these problems by developing a revisiting system that addresses each of the previously stated problems.

1. **There is a lack of integration between revisitation facilities**. I will produce a single integrated system for page revisitation. I will do this by unifying Back, History and Bookmarks into a single interface, displayed in a single window, and all based on the same underlying model.
2. **There are identified problems with the stack-based model of the Back/Forward buttons**. I will produce Back/Forward buttons that operate on a recency model. Previous researchers have found that this model reflects how users expect the buttons to work. Once these are implemented, I will verify through a user study that people can use these buttons at least as well as the current stack-based model buttons.
3. **Recognizing particular pages in the history is difficult.** I will investigate the use of thumbnail images and analyze their effectiveness through a user study – comparing user's recognition of thumbnails vs. titles and URL addresses.
4. **Finding particular pages in the history list is difficult**. I will design the History list based on a single recency-ordered list, so that recently visited pages are easily accessed. Also, I will implement dynamic query filters to help the user isolate desired pages based on several search criteria.
5. **Bookmarks have several recognized problems.** I will unify bookmarks into the history list through 'Dog Ears', where pages can be explicitly marked any time or implicitly marked by its visit count. New bookmark items are made prominent and older bookmark items gradually migrate out of focus. The dynamic query filters will allow Dog Ears to be searched by several criteria.
6. **Many alternatives to the standard revisiting mechanisms require heavy use of screen real estate**. I will consider only designs that are constrained by the screen real estate they use. In particular, the design will use roughly one quarter of the browser window width, comparable to the standard History bar provided in Internet Explorer.

1.4 Thesis Outline

In Chapter 2 I will discuss the usability issues present in conventional web browsers and illustrate how alternative browsers have tried to address them. In Chapter 3 I will present the history system I designed and implemented as part of this thesis. I will also discuss how this design addresses each of the identified usability issues. Chapter 4 describes the user study I ran to determine how well people can identify previously seen web pages when shown the page’s thumbnail image, title or URL. In Chapter 5 I discuss the technical issues that arose in implementing my history system. Finally, Chapter 6 describes the extent that I have met the goals described in Section 1.3.

2. Issues for revisitation facilities

In the previous chapter, I outlined usability problems found in today's current revisitation facilities.

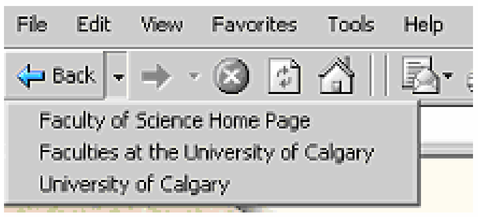
1. There is a lack of integration between revisitation facilities.
2. There are identified problems with the stack-based Back/Forward buttons.
3. Recognizing particular pages in the History list is difficult.
4. Finding particular pages in the History list is difficult.
5. Bookmarks have several recognized problems.
6. Many alternatives to the standard facilities require heavy use of screen real estate.

In this chapter, I will describe each problem in detail. Although I will show examples of these problems in two prevalent browsers, Netscape Navigator and Microsoft's Internet Explorer (referred to as IE), these problems are generally common to all commercial web browsers[1]. I will also illustrate alternative browsers that have tried to address particular problem aspects.

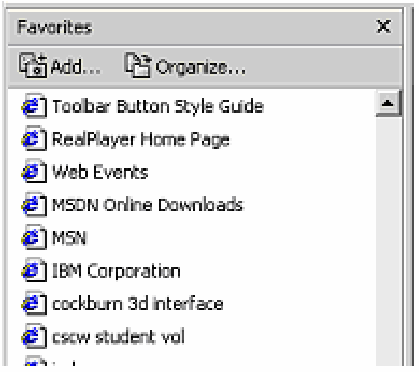
2.1 Lack of integration between facilities

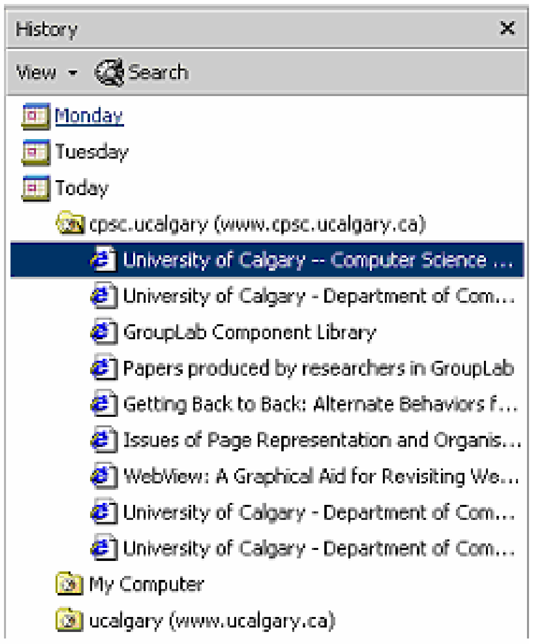
The standard web browsers provide many different mechanisms for revisiting pages—including the Back/Forward buttons, History list and the Bookmark list. These facilities are completely separate and have competing interfaces, each with different rules about which pages they offer to the user, how they display pages, and what operations the user can perform on them.

[1] This thesis will refer to Netscape Navigator versions 4 and 6, and Internet Explorer 4 through 5.5.


File
Edit
View
Favorites
Tools
Help
Back
Faculty of Science Home Page
Faculties at the University of Calgary
University of Calgary

a) Back/Forward Menu




History
View
Search
Monday
Tuesday
Today
cpsc.ucalgary (www.cpsc.ucalgary.ca)
University of Calgary -- Computer Science ...
University of Calgary - Department of Com...
GroupLab Component Library
Papers produced by researchers in GroupLab
Getting Back to Back: Alternate Behaviors f...
Issues of Page Representation and Organis...
WebView: A Graphical Aid for Revisiting We...
University of Calgary - Department of Com...
University of Calgary - Department of Com...
My Computer
ucalgary (www.ucalgary.ca)

b) Bookmarks c) History List

Figure 2-1. Internet Explorer's separate Back, Bookmarks (Favorites), and History facilities.

Figure 2-1 illustrates how the user can raise different listings for Back/Forward, Bookmarks (called 'Favorites' in Microsoft's IE) and History. Although these screenshots were taken at the same time, each list has entirely different pages and representations for these pages.

Figure 2-1a shows the pages available in the Back button dropdown list. The only actions available to the user are to select an item from this list to navigate to that page, or to click Back to stepwise navigate to that page. Figure 2-1b displays the user's bookmarks; items can be added, rearranged, renamed and deleted through various mouse and menu commands. Finally, Figure 2-1c illustrates the history listing. Here items can be accessed, deleted, but not reordered or renamed (although other views are possible).

The user is expected to know not only the rules and operations available in each list, but also which listing will provide the easiest access to a desired page, as there are situations where the page can be found in all of them.

Splitting up these facilities increases not only the user's cognitive decision-making burden, but also the physical effort needed to carry out tasks. Additional mouse movements and clicks are required to raise/hide the different facilities, to drag items from the history list into the bookmark list, and to scroll through the lists. While users are interested in finding and accessing previously seen web pages, they are forced to deal with pages repeatedly as either Back, History or Bookmark items.

2.2 Problems with stack-based Back/Forward

The Back button is the most utilized revisitation mechanism. Catledge and Pitkow (1995) found it accounted for 41% of all users web browser actions, second only to clicking on hyperlinks. Tauscher and Greenberg (1997) found similar results, as it accounted for 35% of all actions, again trailing only hyperlinks. Despite this high usage, however, studies find that people have naïve models about how Back actually works and what pages can be reached through it. Cockburn and Jones (1996) asked 11 computer professionals about the back button, and found that only one knew about its stack model. The other ten incorrectly thought that it operated as a simple recency-ordered list.

I will now explain the stack model, and then propose how a recency model addresses some of the current system's problems.

2.2.1 Stack model

To explain this stack concept, we will follow the example illustrated in Figure 2-2. As the user navigates to pages *a–e* (Figure 2-2a, left), each of these pages is added to the stack. The stack has a stack pointer (shown as black arrow in Figure 2-2a) that points to the page that the user is currently looking at. The user now decides to move back to page *c* by clicking the back button twice. Figure 2-2b illustrates how this action moves the stack pointer down to page c. Now the user navigates to a new page *i*. Figure 2-2c illustrates how the browser pops pages d and e off the stack. Finally, Figure 2-2d illustrates the resulting stack. The stack no longer has entries for pages *e* and *d*. Thus, no matter how many times the user clicks Back, these pages will never appear.

The stack model only allows users to access pages on a single thread of navigation. Thus, users can become frustrated when trying to backtrack to a recently viewed page, only to find it has mysteriously disappeared from the list.

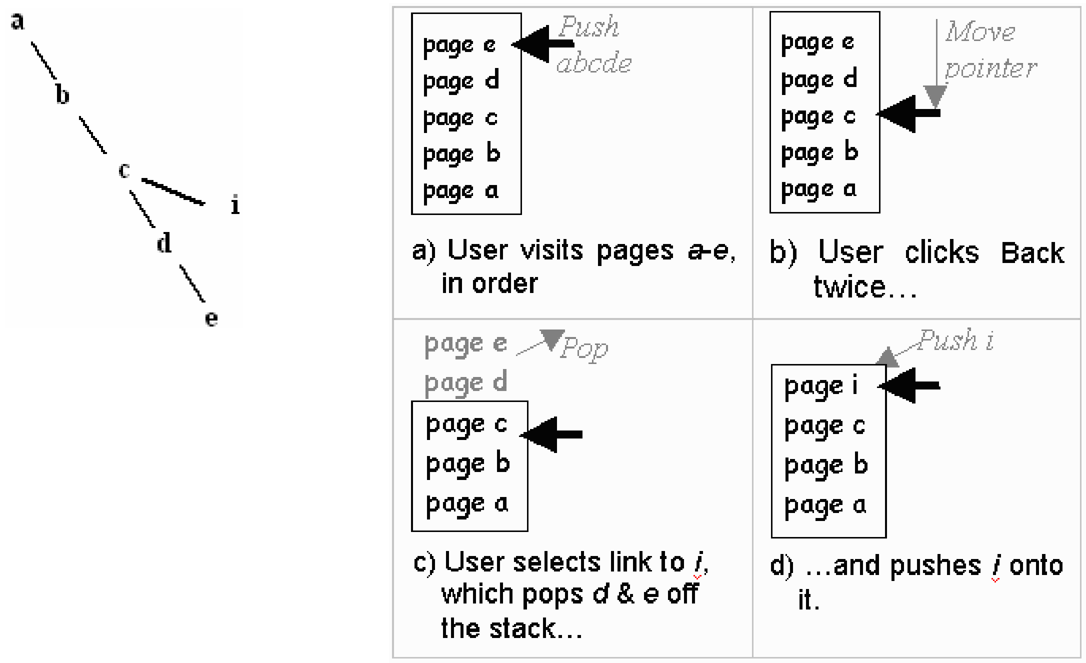

a
b
c
d
e
i
page e
page d
page c
page b
page a
Push abcde
a) User visits pages a-e, in order
Move pointer
b) User clicks Back twice...
Pop
c) User selects link to i, which pops d & e off the stack...
Push i
page i
d) ...and pushes i onto it.

Figure 2-2. An example navigational trace and its effect on the stack. Note that previously visited pages d and e are no longer available on the stack (reproduced from Greenberg and Cockburn, 1999).

In simple cases, this may not be a concern, as the user can retrace their steps to return to these pages. To return to pages *d* and *e*, the user can use Back to return to page *c* and then revisit the links to page d and after that e. Greenberg and Cockburn (1999) suggest this is only reasonable for short pages with few links and a short navigational path. Some pages have many links or a complex navigation route, making it difficult to retrace steps to find a particular page

2.2.2 Recency model

Greenberg and Cockburn (1999) have proposed several alternate behaviours for the back/forward buttons. One of these models, termed, *Recency with Temporal Ordering Enhancement* ensures all pages can be revisited, and that they appear in their precise recency ordering. Figure 2-3 illustrates an example of this model.

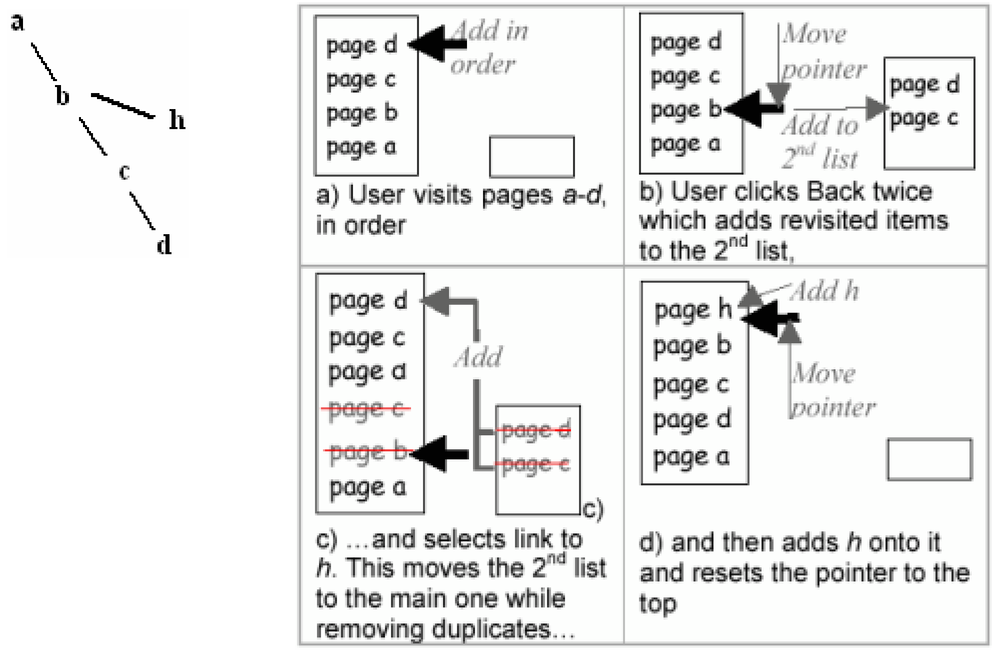

a
b
h
c
d
page d
page c
page b
page a
Add in order
a) User visits pages a-d, in order
page d
page c
page b
page a
Move pointer
Add to 2nd list
page d
page c
b) User clicks Back twice which adds revisited items to the 2nd list,
page d
page c
page a
~~page c~~
~~page b~~
page a
Add
~~page d~~
~~page c~~
c)
c) …and selects link to h. This moves the 2nd list to the main one while removing duplicates…
page h
page b
page c
page d
page a
Add h
Move pointer
d) and then adds h onto it and resets the pointer to the top

Figure 2-3. An example using Recency-based back/forward buttons (Greenberg, Ho, Kaasten, 2000).

First, the user visits pages *a–d*. Figure 2-3a illustrates how the web browser adds these pages to the list, so that the stack pointer is pointing to page d. Next, the user presses Back twice to return to page *b*. Figure 2-3b illustrates how this action moves the list pointer to point to page *b*, and adds pages *d* and *c* to a secondary list. The user now navigates to new page *h*. Since the user navigated without pressing the Back/Forward button, the web browser now sorts the list to reflect the proper recency ordering. Figure 2-3c illustrates how the web browser moves the items in the secondary list to the main list, removing duplicates. Finally, the new page h is added to the top of the list. Figure 2-3d illustrates this final state, where the list is sorted by recency of the user's last visit.

This model provides two important guarantees. First, all of the visited pages are in the list, so that the user can reach any page by clicking Back. Second, the pages appear in the order that the user last saw them.

Greenberg and Cockburn (1999) note that this behaviour introduces extra work for the user in some situations. Tauscher and Greenberg (1997) found that users often navigate in a 'hub and spoke' manner. That is, they start at a central page, and then visit many different 'spoke' pages each from the 'hub'. The stack-based Back button makes it easy for the user to jump between 'hub' pages, as the spoke pages are cleared off the stack. The recency-based Back button includes all spoke pages, so if the user is only interested in hub pages he or she must click the back button many more times.

Table 2-1 summarizes the advantages and disadvantages of the stack and recency model for Back/Forward.

Table 2-1. Comparison of Stack and Recency models.

Model	**Advantage**	**Disadvantage**
Stack-based back/forward	Automatic page pruning may remove 'spoke' pages that are no longer needed. E.g. When the user wants to move back to previous 'hub' pages.	Cannot return to pages that have been popped off the stack. This comes as a surprise to users.
Recency-based	All visited pages are accessible via back/Forward. Pages reflect Recency ordering.	Since duplicates are removed, users might not recognize the recency ordering. Since there is no pruning, more clicks are needed to move up a hierarchy than in the stack-model.

Having explained the models, it is important to address their actual use. A user study by Greenberg, Ho, Kaasten (2000) compared subjects' preferences between the stack-based and Recency-based Back. The subjects carried out several typical web tasks using each model. The study confirmed Cockburn and Jones' (1996) finding that users have a naïve mental model about how Back works. The experiment's subjects were evenly split between preferring the Stack model and the Recency model. This suggests that users can adapt to a recency model of Back, of course assuming the change is merited. The complete study is included as Appendix 1.

2.3 Page recognition in history lists

For a history mechanism to be useful, the user must be able to find and recognize its pages quickly with minimal effort. This falls upon both the history list's representation of individual pages and the way that these pages are organized.

The standard web browser history list relies on textual representations of web pages—namely the page title or its URL. Unfortunately, this is not always how users remember the page.

2.3.1 Title

Titles are the caption taken from the <title> tag in the document's source. Cockburn and Greenberg (1999) have found page titles are often problematic for several reasons. First, many pages do not contain a title tag. Cockburn and McKenzie found this to be the case for about 5% of their subjects' pages. My history study, described in Chapter 5, found that 30% of users' pages do not have titles. This difference is due to how my study included unfiltered user history files (See Section 4.3.5). Even if only 5% of the user's pages are untitled, the issue remains that these pages must be represented another way. When there is no title, most history systems will substitute the page's URL in the titles place, while others (e.g., Netscape 4, shown in Figure 2-4, see item 7) leave the field blank.

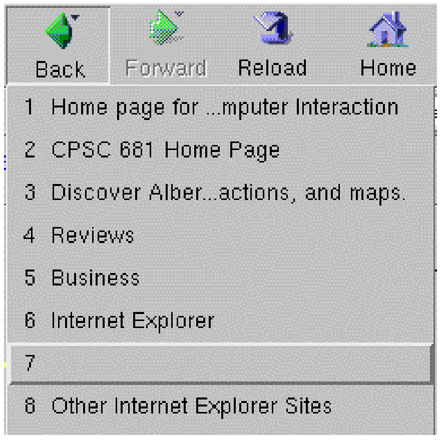

Back
Forward
Reload
Home
1 Home page for ...mputer Interaction
2 CPSC 681 Home Page
3 Discover Alber...actions, and maps.
4 Reviews
5 Business
6 Internet Explorer
7
8 Other Internet Explorer Sites

Figure 2-4. Netscape's Back menu, showing page titles (taken from Cockburn and Greenberg, 1999).

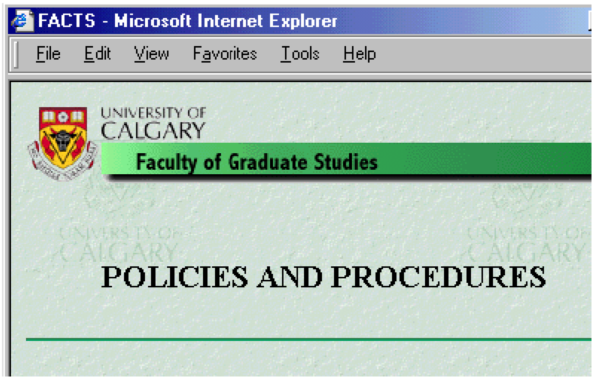
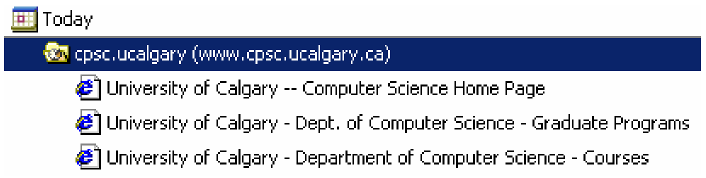

FACTS - Microsoft Internet Explorer
File Edit View Favorites Tools Help
UNIVERSITY OF CALGARY
Faculty of Graduate Studies
POLICIES AND PROCEDURES

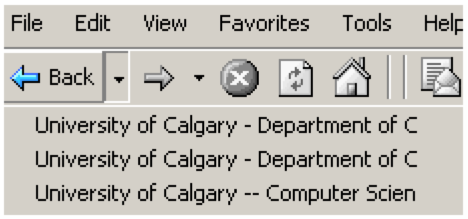
Figure 2-5. Page title (Facts) differs from actual page content.

Often, there is a mismatch between the title and actual page contents. First, titles do not reflect or summarize the content of the page. We believe this is because many page creators use a previously created page as a template, but forget to change the page's title property to reflect the new content. Figure 2-5 is an example extracted from the University of Calgary web site, where the page title, 'Facts' (shown at very top of window border) is quite different from the actual page heading, 'Policies and Procedures' (shown in document). Second, some web sites use the same title for all of its pages. While the user can easily identify the web site they belong to, it is impossible to tell such pages apart by their title. Third, web page titles can often be quite long. Yet, web pages with descriptive, well-formed titles are often too long to fit into the history mechanism's confined space. Figures 2-6 and



Today
cpsc.ucalgary (www.cpsc.ucalgary.ca)
University of Calgary -- Computer Science Home Page
University of Calgary - Dept. of Computer Science - Graduate Programs
University of Calgary - Department of Computer Science - Courses

Figure 2-6. IE's history list widened so that long titles can fit.



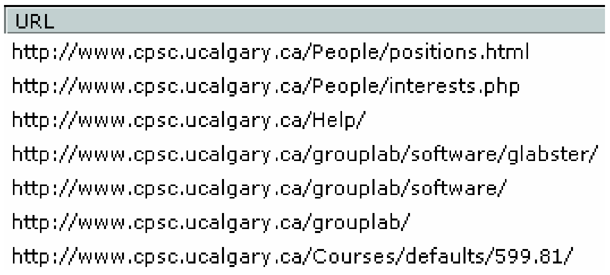
University of Calgary - Department of C
University of Calgary - Department of C
University of Calgary -- Computer Scien

Figure 2-7. IE's Back/Forward button list is not resizable, so the titles appear truncated.

2-7 show how the usefulness of page titles depends greatly on the size available to display them. Again, these are examples from the University of Calgary web site. We stress that this type of labelling is not unusual.

2.3.2 URL

Like titles, URLs have various problems that make them difficult to recognize. First, URLs are often perplexing because they refer to their technical location on the Internet rather than their actual content. While the URLs for major sites, like www.microsoft.com and www.netscape.com, are recognizable to many, the URLs for the smaller web pages, like sern.ucalgary.ca/~kaasten don't have the same recognition, and offer little clue about the content of the page.

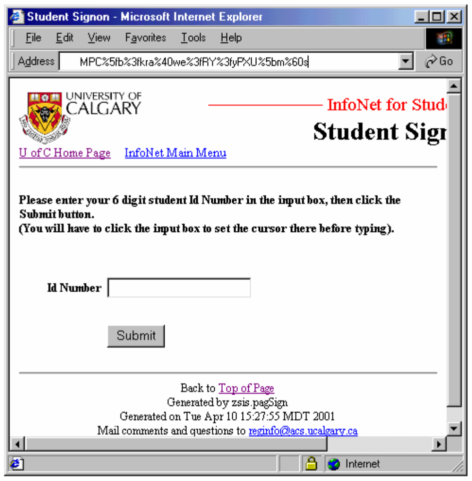


URL
http://www.cpsc.ucalgary.ca/People/positions.html
http://www.cpsc.ucalgary.ca/People/interests.php
http://www.cpsc.ucalgary.ca/Help/
http://www.cpsc.ucalgary.ca/grouplab/software/glabster/
http://www.cpsc.ucalgary.ca/grouplab/software/
http://www.cpsc.ucalgary.ca/grouplab/
http://www.cpsc.ucalgary.ca/Courses/defaults/599.81/

Figure 2-8. URL history list in Netscape 6.

Second, the size requirement to display URLs is often greater than for titles. Figure 2-8 shows reasonable URLs, many of which require a field more than 45 characters wide. By default, Internet Explorer displays approximately 35 characters.

Third, it is increasingly common to have cryptic URLs generated automatically by computers rather than by a person. This is especially true for search engines and web-based applications. For example, Figure 2-9 illustrates a web page taken from the University of Calgary's online registration system. Next to the figure is the URL that for that page.


Student Signon - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address
Go
UNIVERSITY OF CALGARY
Submit
Internet

MPC%5fb%3fkra%40we%3fRY%3fyPXU%5bm%60s

InfoNet for Stud

Student Sign

U of C Home Page InfoNet Main Menu

Please enter your 6 digit student Id Number in the input box, then click the Submit button.
(You will have to click the input box to set the cursor there before typing).

Id Number

Back to Top of Page
Generated by zsis.pagSign
Generated on Tue Apr 10 15:27:55 MDT 2001
Mail comments and questions to reginfo@acs.ucalgary.ca

https://dciswp.admin.ucalgary.ca/cgi_bin/ndCGI.exe/zsis_menu?SPIDERSESSION=FF%7cqT%5ddqyTXqApoH%5f%40KzBFJuCrEXVAK%5bFwoE%5dG%5bVK%5f%5b%60fPs%60mpOXFpq%5b%5bdZh%5bDXq%3f%5dcuCy%3fa%40V%7cryRsWgACDLqAy%5bjNh%5bdlq%5bmdQm%7ddqEY%5dw%7efIryVIhbWsUBBnA%5biY%60mNw%5bGwo%5eAPoR%5f%3f%5bqfQCpCtqX%3fQIGFf%5fEDw%5bzHWs%3fMPC%5fb%3fkra%40we%3fRY%3fyPXU%5bm%60s

Figure 2-9. Cryptic URL page.

2.3.3 Thumbnail images

Some systems utilize *thumbnail* images, which are images of the web page scaled down to a small size. Neither Internet Explorer nor Netscape have made use of thumbnails[2]. However, alternative research revisitation systems have incorporated thumbnails into their user interface. Two such systems are MosaicG (Ayers and Stasko, 1995) shown in Figure 2-10, and PadPrints (Hightower, Ring, Helfman, Bederson, Hollan, 1998) shown in Figure 2-11.

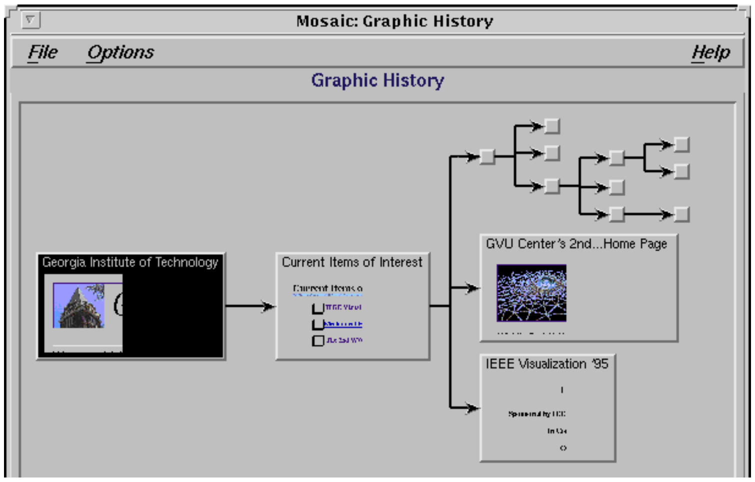

Mosaic: Graphic History
File
Options
Help
Graphic History
Georgia Institute of Technology
Current Items of Interest
GVU Center's 2nd...Home Page
IEEE Visualization '95

Figure 2-10. MosaicG (Ayers and Stasko, 1995).

[2] Microsoft Windows uses thumbnail images in its 'Explorer' File Manager.

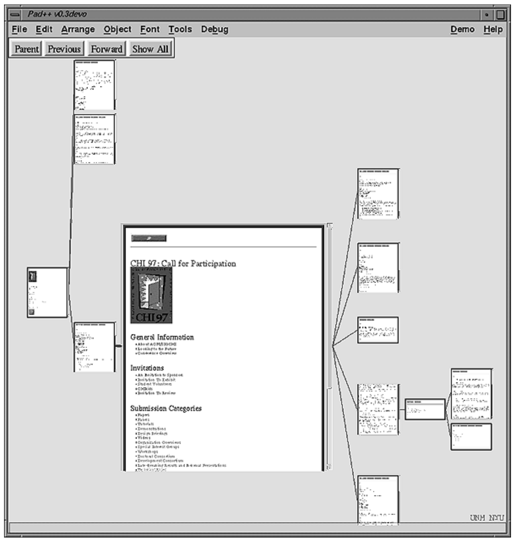

Pad++ v0.3devo
File Edit Arrange Object Font Tools Debug Demo Help
Parent Previous Forward Show All
CHI 97: Call for Participation
General Information
Invitations
Submission Categories

Figure 2-11. PadPrints (Hightower, Ring, Helfman, Bederson, Hollan, 1998).

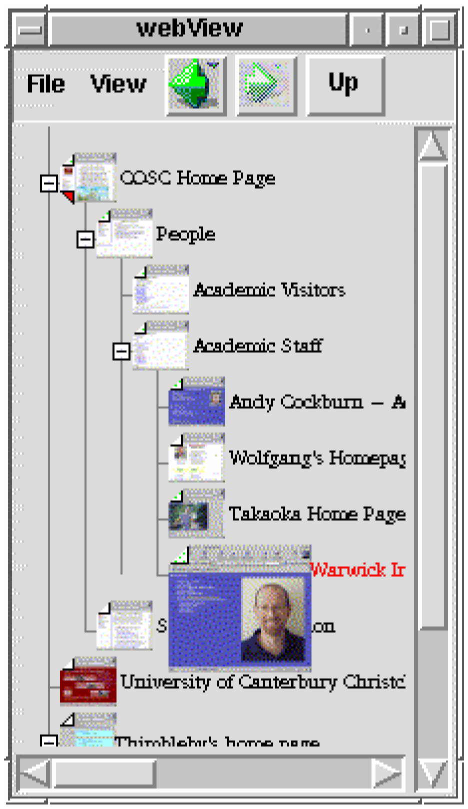


Figure 2-12. WebView (A Cockburn, S Greenberg, B McKenzie, M Smith, and S Kaasten, 1999).

Similarly, Andy Cockburn developed WebView (Cockburn et al, 1999) shown in Figure 12. Cockburn developed WebView during collaborative research between the University of Canterbury and this research at the University of Calgary. Both WebView and my own Unified History (described in Chapter 3) started from the same collaborative intellectual roots, where there was a heavy and mutual exchange of design ideas. Thus, many of the ideas presented in this thesis (thumbnails, Dog Ears, implicit bookmarks) are also present in Cockburn's research.

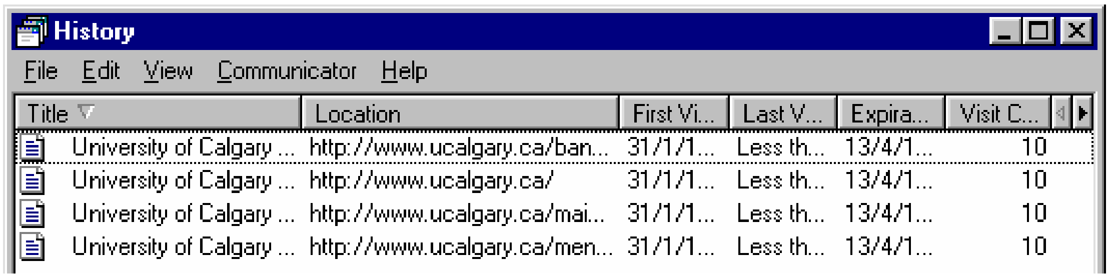
A thumbnail is a direct representation of the page, i.e. how it was seen by the user, only smaller. This makes it a more promising way to represent a page than abstract or possibly erroneous page titles or cryptic URLs.

However, thumbnails are not without their drawbacks, especially if they are scaled to small sizes. For example, Figure 2-13 shows several thumbnail images from four different pages of one web site. The site maintains a consistent graphical look on all of its pages, which makes it difficult to distinguish between their thumbnails at this small size. Also, the text appearing in the small image is hard or impossible to read.

Figure 2-13. Four different pages within a web site, viewed at 50 pixels wide (reproduced from Cockburn and Greenberg, 1999).

2.4 Page organization

No matter how well pages are represented in the history list, finding a particular item can be difficult when the list is lengthy. Thus, how pages are ordered in the list is an important usability issue. There are many ways pages can be ordered. For example, Netscape's history list, shown in Figure 2-14, allows the user to choose how pages are ordered by clicking particular columns: by page title, URL, date of first visit, date of last visit or visit count. Yet flexibility of choice does not necessarily mean pages can be found easier.



Title	Location	First Vi...	Last V...	Expira...	Visit C...
University of Calgary ...	http://www.ucalgary.ca/ban...	31/1/1...	Less th...	13/4/1...	10
University of Calgary ...	http://www.ucalgary.ca/	31/1/1...	Less th...	13/4/1...	10
University of Calgary ...	http://www.ucalgary.ca/mai...	31/1/1...	Less th...	13/4/1...	10
University of Calgary ...	http://www.ucalgary.ca/men...	31/1/1...	Less th...	13/4/1...	10

Figure 2-14. Netscape History List.

2.4.1 Sorting alphabetically by URL

Sorting the list by URL is useful for grouping together pages by web site. However, this method is not foolproof, as seemingly related web pages may span more than one site. For example traveling from Microsoft's homepage to their travel service site involves following just three hyperlinks, but this trip actually spans three different sites— from http://www.microsoft.com, to http://www.msn.com/, and finally to http://expedia.com. These three pages, while linked directly to each other, would be spaced far apart in the URL sorted listing. Even if this were not an issue, recognizing pages by URL has problems as already mentioned.

2.4.2 Sorting alphabetically by title

The problems with recognizing titles described in 2.3.1 also apply to list organization. If the list contains pages with poor titles, sorting the list by these titles will not make it easy to find pages. However, even pages that have reasonable titles will be difficult because the user needs to know the exact title. For example, while one might expect the Microsoft homepage to be in the ‘m’ section, the page title is actually, “Welcome to Microsoft’s Homepage”, so it would be found with the ‘w’s.

2.4.3 Sorting by first visit

Sorting the list by date of visit has the advantage that the list order reflects the user’s actions. Thus, users can predict where items will appear. However, having items sorted by first visit has two problems. First, items remain in their original context, i.e., surrounded by the pages that the user also visited before and after that page. This context may not be relevant later. Second, when the user revisits a page, it stays at the bottom. Thus, pages that the user has recently visited will be at the bottom of the list, requiring the user to scroll down to the bottom to find them. Greenberg (1993a) observed in other domains that sorting by original instance is a poor way of predicting future actions.

2.4.4 Sorting by Last Visit (Recency)

Recency has been found to be a valuable history pattern in several domains, as reported by Greenberg (1993a). Greenberg found that 41% of phone calls are repeats of one of the previous ten. His study of Unix command lines revealed that there is a 47% chance that the user’s next command is one of the previous ten. Tauscher and Greenberg (1997) found that recency is just as valuable in web page revisiting. They found that while revisiting accounts for 58% of all web navigation, revisiting to one of the previous ten pages accounts for 43% of navigation. Thus, using this sorting, there is a 43% chance that the user’s next web page will be one of the top 10 pages in the list. Clearly, recency is an extremely useful method for sorting a history list. As described later in this thesis, I will use recency sorting within my new revisitation system.

2.4.5 Sorting by Visit Count (Frequency)

Overall, Tauscher and Greenberg (1997) found frequency to be a mediocre predictor of web page revisiting. Many recently revisited pages are only seen a few times, and thus will not appear in a top position on the frequency list. Also, some frequently visited pages go 'stale' when they are no longer needed, yet they still occupy a top spot on the list.

However, they found that users have a few key pages that are revisited frequently, that cannot be predicted by merely looking at recency of visits. These pages are often an organization's home page or a search engine. Thus, frequency can be a useful indicator for determining the 'important' pages. However, Tauscher and Greenberg (1997) note that users were sometimes surprised to see which pages they had visited frequently.

2.4.6 History searches

When organization fails, users have the option of searching the history list for pages. Both Netscape's and Internet Explorer's history list have search features. While this system seems to cover all possible needs, there is a high cost in discovering this capability. For example, Netscape's search feature involves raising the history list, opening the search window, specifying a search query and viewing the results, requiring no less than three additional windows. As Greenberg and Tauscher (1997) found in their study, users are unlikely to invest such effort to uncover this functionality.

2.4.7 Combination of sortings

Internet Explorer's history list, shown in Figure 2-15, uses several criteria for organizing the list. The list is organized into a hierarchy, which is sorted by date and then URL.

There are folders for each day of the current week, and single folders for each of the previous weeks. When the user selects one of these folders ('Today' in Figure 2-15), the URL subfolders appear. We suspect that people will have a hard time remembering exactly when they had seen desired page, and would often just guess at these folder dates.

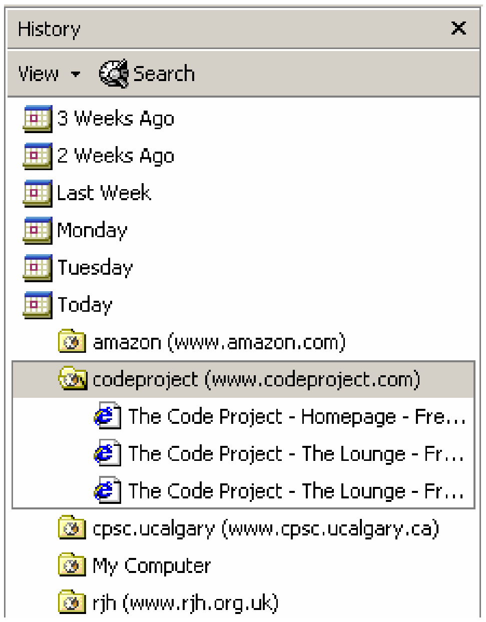

History
View Search
3 Weeks Ago
2 Weeks Ago
Last Week
Monday
Tuesday
Today
amazon (www.amazon.com)
codeproject (www.codeproject.com)
The Code Project - Homepage - Fre...
The Code Project - The Lounge - Fr...
The Code Project - The Lounge - Fr...
cpsc.ucalgary (www.cpsc.ucalgary.ca)
My Computer
rjh (www.rjh.org.uk)

Figure 2-15. Internet Explorer History List (grouped by date).

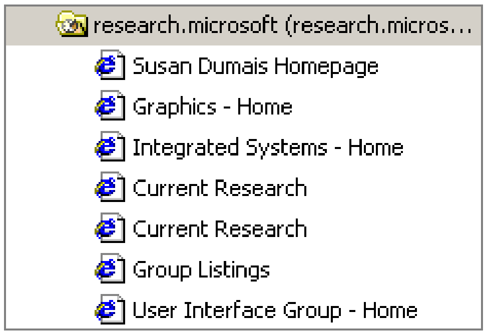

research.microsoft (research.micros...
Susan Dumais Homepage
Graphics - Home
Integrated Systems - Home
Current Research
Current Research
Group Listings
User Interface Group - Home

Figure 2-16. Pages sorted by URLs, though user can only see the titles.

The URL subfolders are sorted by a shortened name that removes the 'www' prefix. In Figure 2-15, there is a folder for 'amazon' rather than 'www.amazon.com'. When the user chooses a URL subfolder (codeproject in Figure 2-15), the pages appear that were visited from that URL domain.

The domains folders are then sorted alphabetically by URL, and looking at the Figure 2-15 from this example, one might think that the pages inside are sorted by title. However, as Figure 2-16 shows, the pages are in fact sorted by URL, even though the user can only see the page titles.

2.5 Bookmark difficulties

Since the early days of the first recognized web browser, NCSA Mosaic, web browsers have allowed users to make bookmarks for returning to important web pages. There are however, several serious problems with bookmarks in even today's browsers that have been carried on from the early browsers.

2.5.1 Marking a page is heavyweight

In order to bookmark a page, the user must consciously decide that a page is important. Thus, the user must predict on their page visit that he or she will want to return to that page at a later time. Unfortunately, people cannot always predict what will be important in the future. It is a common frustration that pages only become important when they are no longer easy to find.

Also, the explicit actions involved with bookmarking a page are heavyweight. The user must interrupt the task at hand to locate the web browser's bookmark functionality, optionally decide on a name for the bookmark, and then decide where to file it. Abrams and Baecker's (1997) bookmark study found that most users do not rename or organize bookmarks when they create them. Instead, they make the bookmark as quickly as possible, to limit the distraction from the task at hand.

2.5.2 Infrequent usage

Perhaps the best evidence of problems with current bookmark systems is that people rarely use them to revisit pages. Both Catledge and Pitkow (1995), and Tauscher and Greenberg (1997) found that bookmarks were selected for less than 2% of all navigations. Rather than sift through the large collection of bookmarks, users instead retrace their steps to find the page, even though this requires numerous navigations instead of directly accessing the page. Abrahms and Baecker (1997) found that users make bookmarks more for long-term archival purposes than as short-term shortcuts. Thus, many bookmarks are not visited for several months.

2.5.3 Difficult to keep organized

Despite infrequent use, many users have large bookmark collections. The Georgia Institute of Technology's 1998 survey found that 35% of web users have collections of more than 100 bookmarks (Pitkow, 1998). Thus, it is important that such large bookmark collections be organized for fast recall.

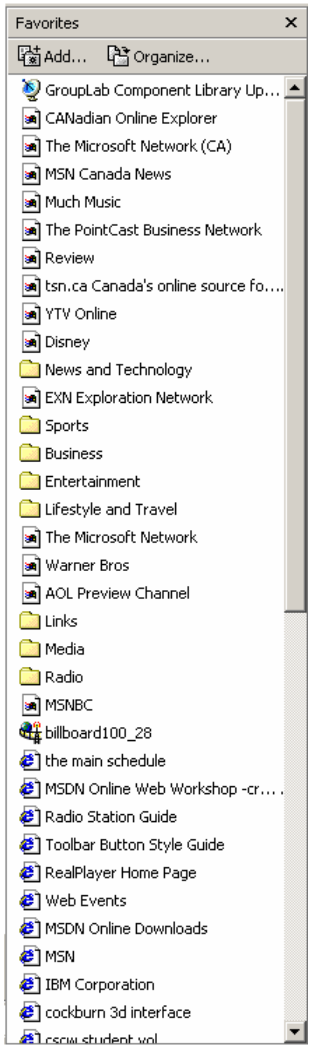


Figure 2-17. IE's Favorites listing.

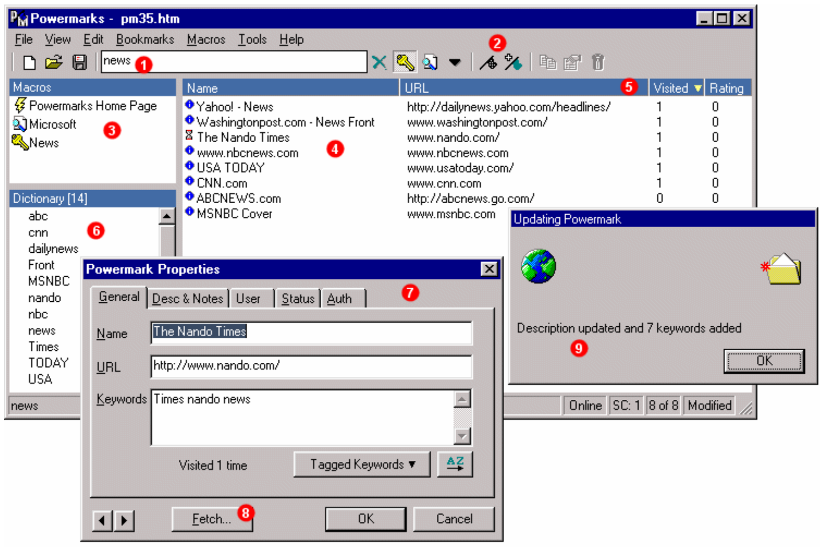
The standard web browsers allow the user to organize bookmarks into folders. In practice, this model breaks down as users accumulate bookmarks while putting off organization. Abrams and Baecker (1997) found that users forgo organizing until the collection is too large to fit in a single screen. Only users with a collection of over 300 items proactively filed bookmarks as they were accumulated.

Bookmark organization also suffers because items are rarely removed. While the standard browsers have several methods for removing bookmarks, users rarely clean out undesired items. Cockburn and McKenzie (2000) found that users are more likely to add new entries than remove no longer needed ones. Indeed, approximately one quarter of a users' bookmark collection point to pages that no longer exist! Figure 2-18 shows how the bookmark list can grow longer than can fit in a single screen (and requires scrolling). To make matters worse, Internet Explorer places new items at the bottom of the list, so that the user must scroll to find the most recently added entries.

Abrams and Baecker (1997) argue that present bookmark facilities require too much effort from the user. These facilities should provide auto-sorting capabilities, based on usage patterns. For example, the listing could be sorted based on recency of use.

2.5.4 Alternative organization: key word retrieval

Recognizing that the folder metaphor requires extensive work from the user, Kaylon Technology's Powermarks explores a new metaphor for a bookmark management utility. It relies on keyword retrieval, allowing bookmarks to be associated with more than one label. Instead of filing a bookmark into a folder, the user associates key words to it. Then, rather than scanning through various folders, the user types in a keyword and dynamically a listing shows all bookmarks that were given that particular keyword. This strategy allows for fast searching. Kaylon pitches “search 20,000 bookmarks in the blink of an eye. Why bother with complicated hierarchies if you can find anything instantly?”



Name	URL	Visited	Rating
Yahoo! - News	http://dailynews.yahoo.com/headlines/	1	0
Washingtonpost.com - News Front	www.washingtonpost.com/	1	0
The Nando Times	www.nando.com/	1	0
www.nbcnews.com	www.nbcnews.com	1	0
USA TODAY	www.usatoday.com/	1	0
CNN.com	www.cnn.com	1	0
ABCNEWS.com	http://abcnews.go.com/	0	0
MSNBC Cover	www.msnbc.com		

Name: The Nando Times
URL: http://www.nando.com/
Keywords: Times nando news
Visited 1 time

Description updated and 7 keywords added

Figure 2-18. Powermarks (Kaylon Technology).

However, assigning key words to bookmarks has the same problems as filing bookmarks into folders. First, the user has to think of words that suitably describe the page. This likely requires as much effort as renaming and filing the bookmark into a folder, which Abrahms and Baecker found users neglected to do. Second, retrieving a bookmark requires raising and switching between several windows. Tauscher and Greenberg (1997) suggest that users are not willing to invest such effort for a ‘helper’ utility.

2.5.5 Alternative organization: spatial organization

Microsoft Research’s Data Mountain (Robertson, Czerwinski, Larson, Robbins, Thiel, van Dantzich, 1998) explores the use of spatial organization for bookmarks. This interface is based on the theory that spatial memory is an effective method for organizing objects. For example, organizing piles of papers on a desk makes it easier to find particular documents later. In this system, the user places bookmarks, represented by thumbnail, on a 3D plane. The plane is inclined (hence the mountain analogy), so that items placed near the top of the screen appear to be more distant than items lower on the screen.

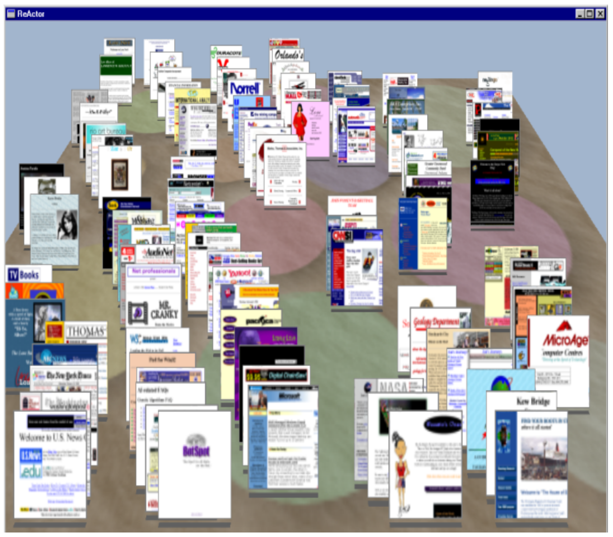


Figure 2-19. Data Mountain (Robertson, Czerwinski, Larson, Robbins, Thiel, van Dantzich, 1998).

This system was not released to the public. Rather, Robertson et al performed a study comparing one group using this system vs. a group using the standard Internet Explorer 'Favorites' system. The experimenters asked each group to organize 100 pre-chosen bookmarks. Thus, the group using the data mountain arranged the bookmarks into groups, or 'piles', while the control group created typical folder hierarchy in the Favorites system. The experimenters then showed the user a cue from one of the bookmarks and timed how long it took the user to retrieve the bookmark from the collection. The study found that user could reliably retrieve bookmarks quickly with the data mountain, even faster than the group using the standard Favorites interface.

However, there are some questions about how this experiment applies to real world use. First, the experimenters chose the pages, rather than using the user's actual bookmark collection. Second, the results depended on the cue shown to the user. The data mountain group was faster at retrieving bookmarks when shown a thumbnail of the desired page. However, there was little difference between data mountain group and the regular IE group when the experimenters only gave the page title. This is expected, as the data mountain displays thumbnails, while the regular Favorites system only displays the page title.

2.6 Screen real estate

Both Internet Explorer 5 (Figure 2-20) and Netscape 6.5 (Figure 2-21) are conservative with screen real estate when they enable the user to view bookmarks in a modest size sidebar that does not cover the main window.

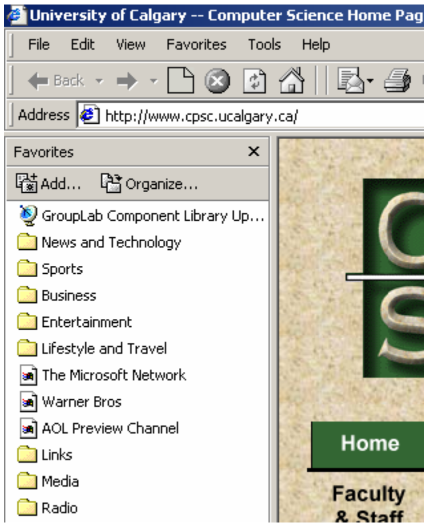


Figure 2-20. IE Favorites sidebar panel.

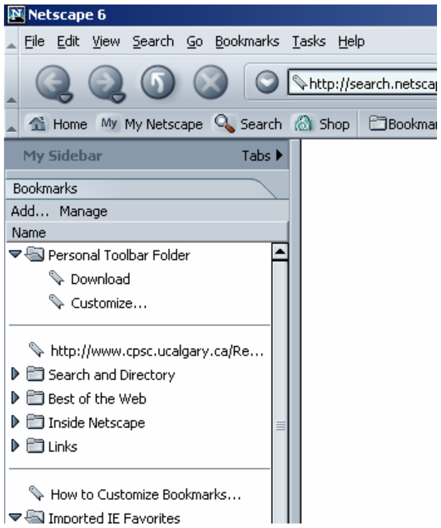
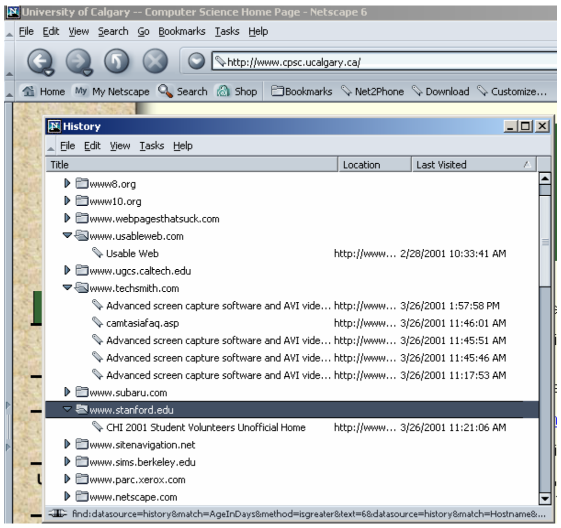


Figure 2-21. Netscape sidebar bookmarks panel.


University of Calgary -- Computer Science Home Page - Netscape 6
File Edit View Search Go Bookmarks Tasks Help
http://www.cpsc.ucalgary.ca/
Home My Netscape Search Shop Bookmarks Net2Phone Download Customize...
History
File Edit View Tasks Help

Title	Location	Last Visited
www8.org		
www10.org		
www.webpagesthatsuck.com		
www.usableweb.com		
Usable Web	http://www...	2/28/2001 10:33:41 AM
www.ugcs.caltech.edu		
www.techsmith.com		
Advanced screen capture software and AVI vide...	http://www...	3/26/2001 1:57:58 PM
camtasiafaq.asp	http://www...	3/26/2001 11:46:01 AM
Advanced screen capture software and AVI vide...	http://www...	3/26/2001 11:45:51 AM
Advanced screen capture software and AVI vide...	http://www...	3/26/2001 11:45:46 AM
Advanced screen capture software and AVI vide...	http://www...	3/26/2001 11:17:53 AM
www.subaru.com		
www.stanford.edu		
CHI 2001 Student Volunteers Unofficial Home	http://www...	3/26/2001 11:21:06 AM
www.sitenavigation.net		
www.sims.berkeley.edu		
www.parc.xerox.com		
www.netscape.com		

find:datasource=history&match=AgeInDays&method=isgreater&text=6&datasource=history&match=Hostname&...

Figure 2-22. Netscape history overlapping main window.

These panels allow the user to select an item and view the resulting page at the same time. Netscape's previous version (6.0) however, implemented the history list as a separate window (Figure 22) that can cover the user's browser window and demands a wide display. Thus, there is not enough room on the user's screen to display both this history list and the main window without overlap.

The alternative browsers that utilize new metaphors for representing and organizing bookmarks and history often require extensive screen space. MosaicG (Figure 2-9), PadPrints (Figure 2-10), Powermarks (Figure 2-15), and the Data Mountain (Figure 2-16), are examples of such systems that require a large portion of the display. Thus, they are often too large to share the same screen as the regular web browser.

Cockburn and Greenberg (1999) maintain that revisiting mechanisms must have both high utility as well as a compact representation to be effective for users. They argue that users will not put forth the effort to raise, move, and hide a revisiting system for routine page revisiting. Tauscher and Greenberg (1997) summarize this point by stating, “It should be cheaper, in terms of physical and cognitive activity, for users to recall URLs from a history mechanism than to navigate to them via other methods.” Users will not adopt a revisitation system that is a burden to operate or competes with their main task.

2.7 Conclusion

While the standard browsers provide support for revisiting, their mechanisms have serious usability issues that limit their convenience. These are listed below.

- Users must learn and switch often between the Back/Forward buttons, History list and bookmark features in order to carry out routine revisiting actions.
- The Back/Forward buttons use a stack model that differs from users’ expectations.
- The representation of pages in the history list can make it difficult to recognize items.
- The ordering of pages in the history list can make it difficult to find items, although recency looks promising.
- Bookmarks require users to know what pages are important in advance, are heavyweight to use, and difficult to keep organized.
- Many alternative revisiting systems require more screen space than users are willing to devote to them.

3. Unified History System

Last chapter I described issues that hinder the revisitation systems in current browsers. In this chapter, I will describe the system I have developed—the Unified History System. I will begin with an overview of the system's design and main features. Then I will describe how Unified History addresses each of the problems highlighted in Chapter 2.

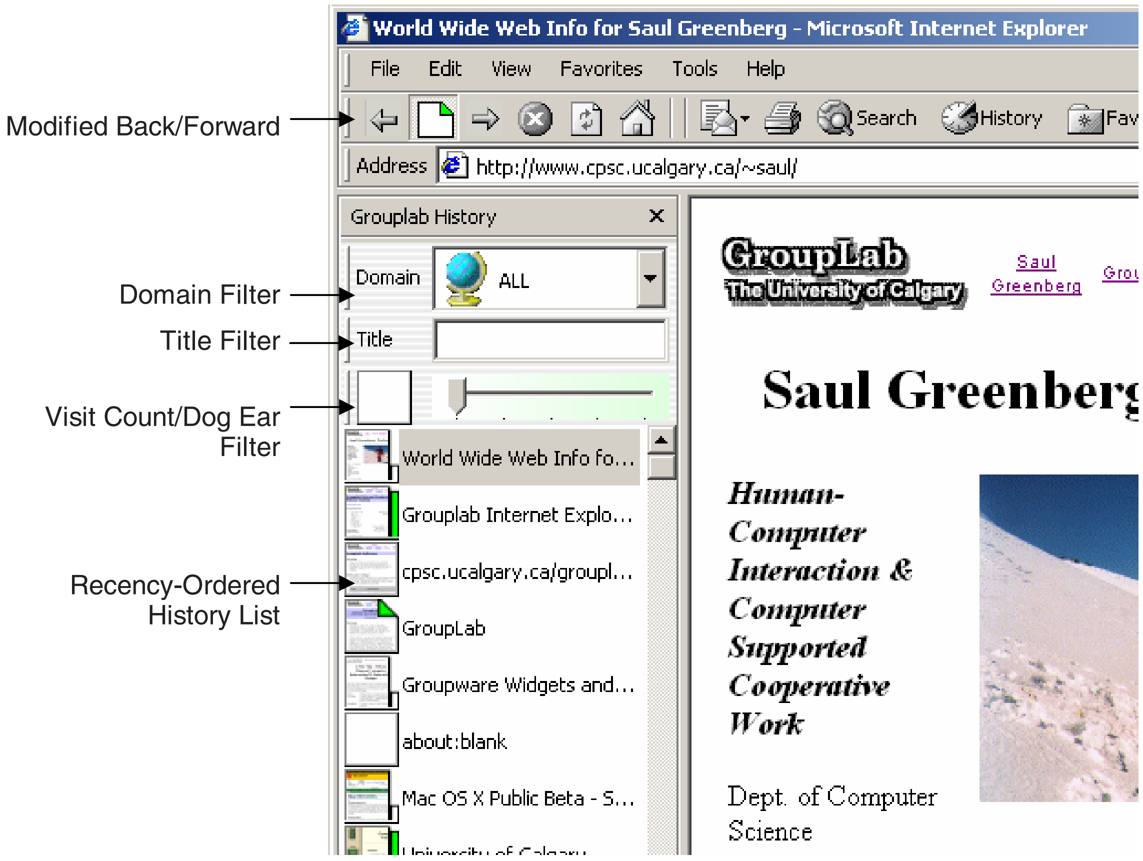

Modified Back/Forward
Domain Filter
Title Filter
Visit Count/Dog Ear Filter
Recency-Ordered History List
World Wide Web Info for Saul Greenberg - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Search History Fav
Address http://www.cpsc.ucalgary.ca/~saul/
Grouplab History
Domain ALL
Title
World Wide Web Info fo...
Grouplab Internet Explo...
cpsc.ucalgary.ca/groupl...
GroupLab
Groupware Widgets and...
about:blank
Mac OS X Public Beta - S...
GroupLab The University of Calgary
Saul Greenberg
Saul Greenberg
Human-Computer Interaction & Computer Supported Cooperative Work
Dept. of Computer Science

Figure 3-1. Unified History features.

3.1 Unified History overview

Unified History has two main visual components: recency-based Back/Forward buttons, and a special History/Bookmark list. Unified History presents these in a way resembling the standard interface practice in IE's revisitation facilities. On the toolbar, Unified History replaces IE's standard stack-based Back/Forward buttons with its own recency-based

Back/Forward buttons. The toolbar also includes a new button for opening/closing the Unified History history list, consistent with IE's standard button for opening/closing its History list. The Unified History History list appears in an explorer bar (the left pane in Figure 3-1), which is how IE presents its own History/Bookmarks and Search add-ins.

3.1.1 The underlying model: recency

Tauscher and Greenberg's (1997) study found that recency with duplicates removed is the best predictor of which pages the user will want to revisit. Thus, all Unified History facilities order the pages by recency, with the most recent pages at the top of the list, where the top 10 entries on this list provide 43% of all user navigation.

3.1.2 Unified History history list

The history list orders all items by recency, with new ones appearing at the top. There are no duplicate items – revisited items move to the top of the list. The history list is optionally displayed when a user clicks the button, and appears as an Explorer Bar window tiled to the left side of the browser (Figure 3-1).

3.1.3 History list page representation: thumbnails, titles and URLs

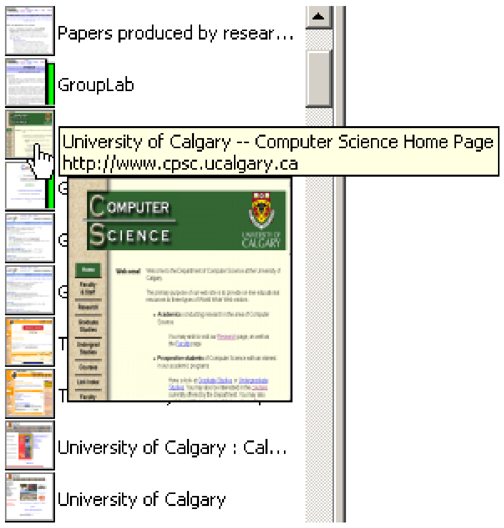

Papers produced by resear...
GroupLab
University of Calgary -- Computer Science Home Page
http://www.cpsc.ucalgary.ca
COMPUTER SCIENCE
University of Calgary : Cal...
University of Calgary

Figure 3-2. Mouse-over tool tip, showing the page's full title, URL and large thumbnail.

Each item identifies its page by a thumbnail image of the actual web page, a (possibly truncated) title, and a pop-up description. When the user moves the mouse cursor over an item in the list, a tool tip shows the full title and URL, along with a zoom-up view of the thumbnail. Figure 3-2 shows the user placing the cursor over the third item, displaying the large thumbnail for that page, its complete title, and its URL. These high quality thumbnails have been scaled down using a smoothing algorithm, discussed in section 5.4. Thus, the large-version thumbnail often contains text that is legible, even at that small size.

3.1.4 Bookmarks as explicit dog-eared pages

While the standard browsers place bookmarks in a separate listing, this system incorporates them into the history list. Bookmarks here are referred to as 'Dog Ears'—pages that have been marked as being important. This concept uses the real life metaphor of folding the corner of a page in a book in order to easily find it again later[3]. These pages have a modified thumbnail to represent this distinction— shown as 🗋—as if the user had actually folded the corner over. In Figure 3-1, the item ' Grouplab' is marked as a Dog Ear, so its icon appears as:

To explicitly mark or unmark an item as a Dog Ear, the user right-clicks on it in the history list. Doing so brings up the context menu shown in Figure 3-3a. At this point the user can type in a new name for the page (Figure 3-3b), or choose to keep the current one by clicking elsewhere. The item is now a Dog Ear, showing a modified thumbnail and title (Figure 3-3c).

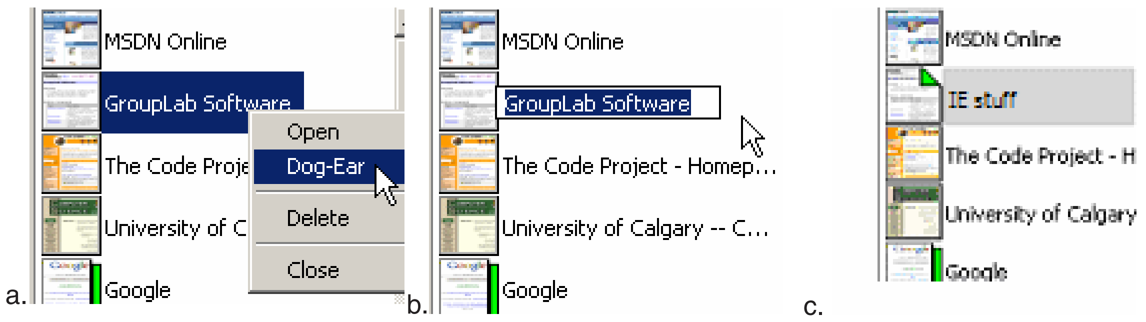
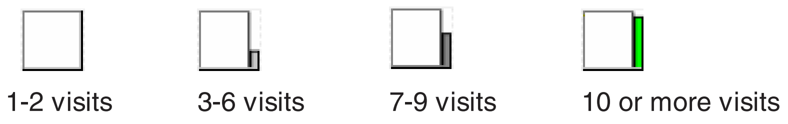


Figure 3-3. Marking an item as a Dog Ear.

[3] DogEars were also used as light-weight bookmarks in Scratchpad (Newfield, Sethi, Ryall, 1998). However, Scratchpad's DogEars are significantly different from ours since theirs do not have a visual representation and do not persist between browsing sessions.

3.1.5 Implicit bookmarks via visit count

Implicit bookmarks visually distinguish a page's importance via the vertical bar to the right of the thumbnail. The higher the bar, the more 'important' a page is judged to be. Currently, importance is estimated by the page's visit count, i.e., how many times the user has visited the page. The possible values and representations are:



The visit-count filter works as a slider with five slots. The first four slots correspond with the above categories. By default, the slider is at the leftmost position, reflected by the □ icon shown next to it, shown in figure 3-4a.

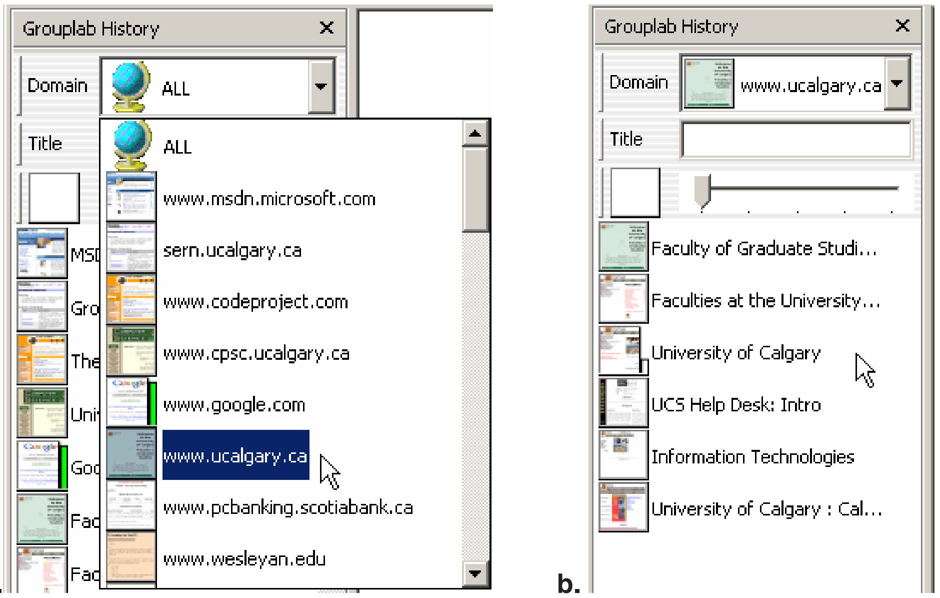
3.1.6 Searching the list via dynamic queries

The entire list contains 58% of all pages a person is likely to visit (as this is the revisitation rate). This can be broken down into 43% occurring within the first 10 items, and 15% appearing further back. To ease searching for these more distant items, several filters are provided, and these are shown above the history listing in Figure 3-1 and in more detail in subsequent figures.

First, the *domain* filter allows the user to see only a selected website that the use has visited. Figure 3-4a shows a user selecting 'www.ucalgary.ca'. From that selection, Figure 3-4b illustrates how the listing is immediately filtered to only include items (still in recency order) from that website.

In keeping with the recency-order theme, the listing of domains is also automatically sorted by recency. Thus, when the user opens the dropdown list, he or she can see the most recent web sites that have been visited. The icon appearing next to each page represents the last web page that the user visited on that site.

The *title* filter works similarly. As the user types in the text box, character-by-character, the listing is updated to only display pages whose title contains that substring. Figure 3-5 shows the changes in the list as the user begins to type 'Grouplab'. By the time the user has typed in ‘gro’, the listing has been reduced from over 60 items to only 10. The outcome of reducing the list can be seen by the changes in the size of the scroll bar. Each character typed reduces the amount of scrolling needed to see the entire list.



a.

Grouplab History

Domain: ALL

Title

- ALL
- www.msdn.microsoft.com
- sern.ucalgary.ca
- www.codeproject.com
- www.cpsc.ucalgary.ca
- www.google.com
- www.ucalgary.ca
- www.pcbanking.scotiabank.ca
- www.wesleyan.edu

b.

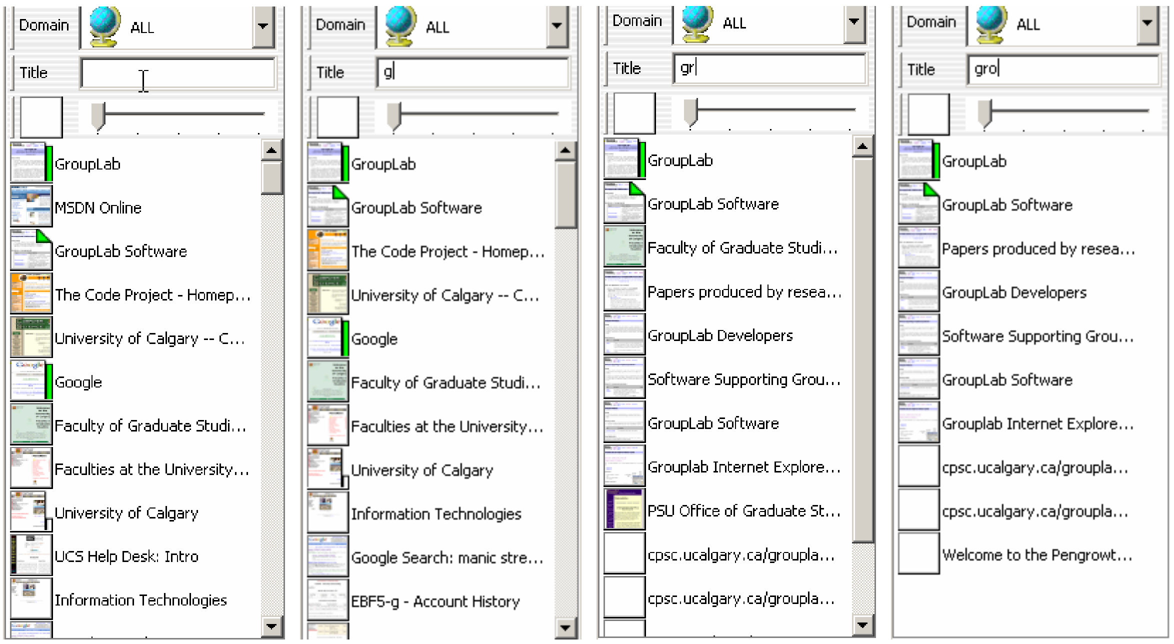
Grouplab History

Domain: www.ucalgary.ca

Title

- Faculty of Graduate Studi…
- Faculties at the University…
- University of Calgary
- UCS Help Desk: Intro
- Information Technologies
- University of Calgary : Cal…

Figure 3-4. Filtering the list to only show pages from the www.ucalgary.ca web site.



Domain: ALL

Title:

- GroupLab
- MSDN Online
- GroupLab Software
- The Code Project - Homep…
- University of Calgary -- C…
- Google
- Faculty of Graduate Studi…
- Faculties at the University…
- University of Calgary
- UCS Help Desk: Intro
- Information Technologies

Domain: ALL

Title: g

- GroupLab
- GroupLab Software
- The Code Project - Homep…
- University of Calgary -- C…
- Google
- Faculty of Graduate Studi…
- Faculties at the University…
- University of Calgary
- Information Technologies
- Google Search: manic stre…
- EBF5-g - Account History

Domain: ALL

Title: gr

- GroupLab
- GroupLab Software
- Faculty of Graduate Studi…
- Papers produced by resea…
- GroupLab Developers
- Software Supporting Grou…
- GroupLab Software
- Grouplab Internet Explore…
- PSU Office of Graduate St…
- cpsc.ucalgary.ca/groupla…
- cpsc.ucalgary.ca/groupla…

Domain: ALL

Title: gro

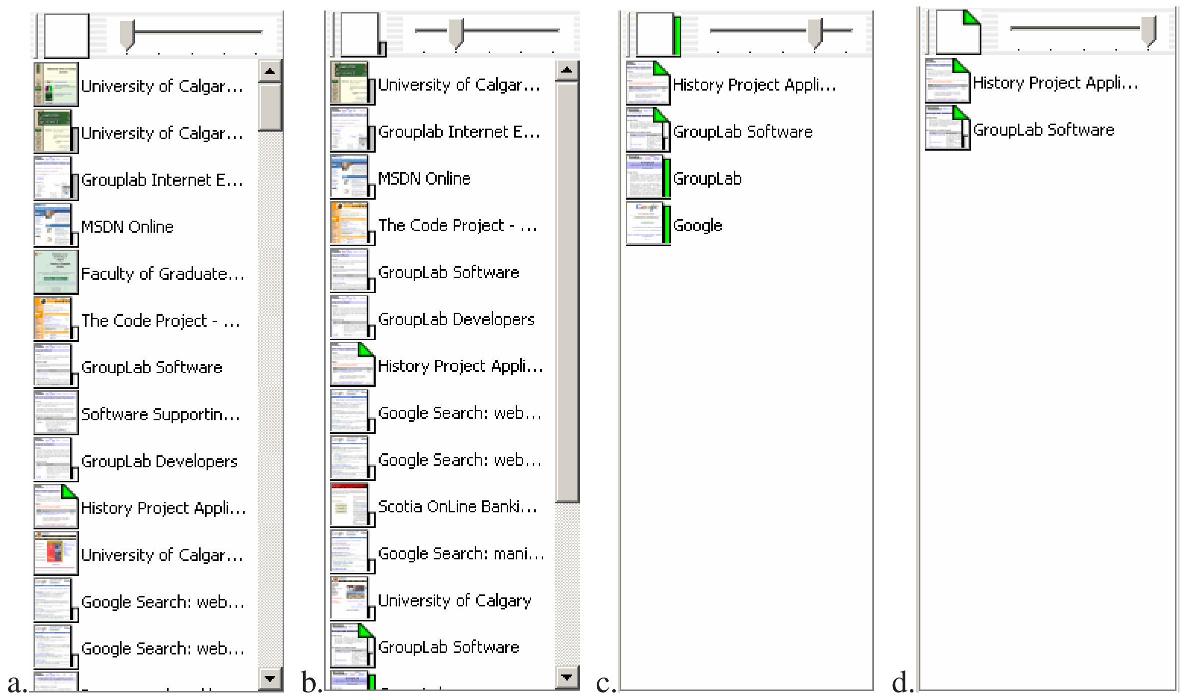
- GroupLab
- GroupLab Software
- Papers produced by resea…
- GroupLab Developers
- Software Supporting Grou…
- GroupLab Software
- Grouplab Internet Explore…
- cpsc.ucalgary.ca/groupla…
- cpsc.ucalgary.ca/groupla…
- Welcome to the Pengrowt…

Figure 3-5. Reducing the history list with the title filter.

The title filter looks at any part of the page titles, not just the beginning. For example, typing ‘soft’ will return a page titled, “Welcome to Microsoft”. Thus, the user does not need to know the exact title of a page, unlike the standard ‘sort by title’ scheme in standard history lists. Instead, a keyword is enough to retrieve the desired page.

The slider filters pages of importance. Here, importance is gauged by two ways. If the page is a Dog Ear, then the page has the regarded as important. Otherwise, the importance is gauged by how often the user has visited the page–the more visits, the more it is deemed to be important.

When the user moves the slider to the next slot, the icon changes to in Figure 3-6b, followed by for the subsequent slot (not shown), and then for the fourth slot in Figure 3-6c. The slider icon indicates that the list has been filtered to only display pages that have been visited the specified number of times. Thus, this filter changes the listing from having all pages shown in Figure 3-6a, to only pages that have been visited more than 10 times in Figure 3-6c.



a.
University of Calgar...
University of Calgar...
Grouplab Internet E...
MSDN Online
Faculty of Graduate...
The Code Project - ...
GroupLab Software
Software Supportin...
GroupLab Developers
History Project Appli...
University of Calgar...
Google Search: web...
Google Search: web...

b.
University of Calgar...
Grouplab Internet E...
MSDN Online
The Code Project - ...
GroupLab Software
GroupLab Developers
History Project Appli...
Google Search: web...
Google Search: web...
Scotia OnLine Banki...
Google Search: mani...
University of Calgary
GroupLab Software

c.
History Project Appli...
GroupLab Software
GroupLab
Google

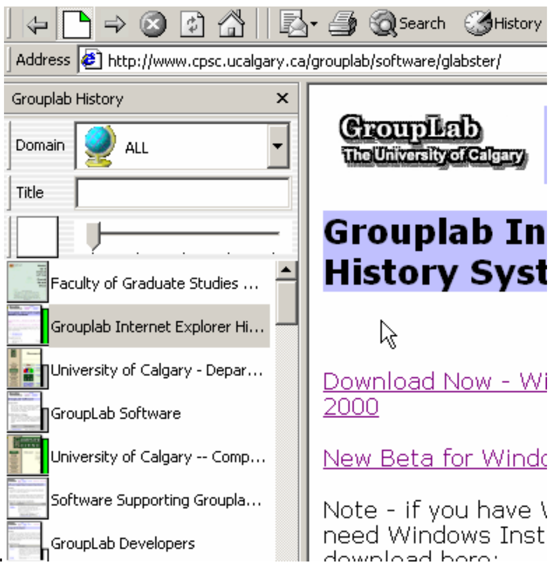
d.
History Project Appli...
GroupLab Software

Figure 3-6. Reducing the history list with the visit count/Dog Ear filter.

However, this slider does not filter off Dog Ear pages as these pages are deemed to be the most important. The fifth and final setting on the slider filters off all pages except for the Dog Ear pages. The slider icon appears as 🗋 to reflect this, shown in Figure 3-4d. Filtering the list by this setting effectively turns the history list into a recency-sorted bookmark listing.

3.2 Back/Forward buttons

When the user installs the system, the toolbar contains new Back and Forward buttons. These buttons operate directly on the unfiltered history list, using the *Recency* algorithm (explained in Section 2.2.2) rather than using the stack method. The Back/Forward button action simply moves down/up the history list, where the current page that the user has open is highlighted in the listing, shown in Figure 3-7a. This highlighting serves as an index, showing the user where the current page fits in the context of the history of navigations. Pages below the highlighted pages can be reached by pressing Back and pages above the highlighted page can be reached by pressing Forward.   In Figure 3-7b, the user has clicked Back so that the current page is now one below the previous one. Unlike the stack model buttons, the user can reach any page in the history by clicking Back or Forward. Also, unlike stack, Back/Forward are merely interface shortcuts for navigating the history list.

a.

Search
History
Address http://www.cpsc.ucalgary.ca/grouplab/software/glabster/
Grouplab History
Domain
ALL
Title
Faculty of Graduate Studies ...
Grouplab Internet Explorer Hi...
University of Calgary - Depar...
GroupLab Software
University of Calgary -- Comp...
Software Supporting Groupla...
GroupLab Developers
GroupLab
The University of Calgary
Grouplab In
History Syst
Download Now - Wi
2000
New Beta for Windo
Note - if you have
need Windows Inst

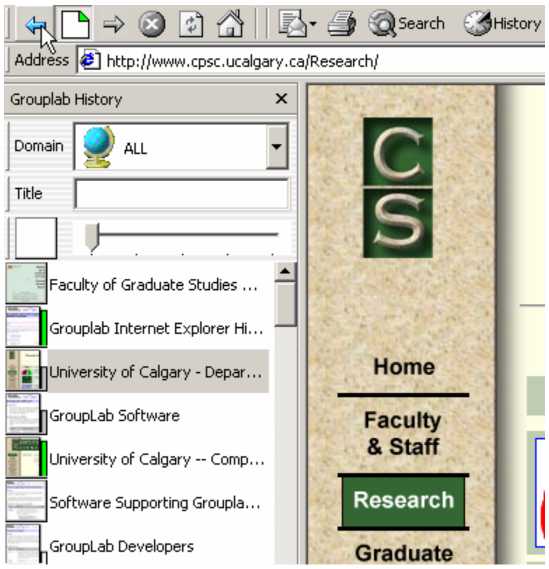
b.

Search
History
Address http://www.cpsc.ucalgary.ca/Research/
Grouplab History
Domain
ALL
Title
Faculty of Graduate Studies ...
Grouplab Internet Explorer Hi...
University of Calgary - Depar...
GroupLab Software
University of Calgary -- Comp...
Software Supporting Groupla...
GroupLab Developers
CS
Home
Faculty & Staff
Research
Graduate

Figure 3-7. Highlighted item in history listing is the index for the back/forward buttons.

Filtering the list does not affect Back/Forward behaviour. Clicking Back or Forward will always navigate to the previous or next page, even if that page has been filtered off the list.

3.2.1 Global History

Since the Back/Forward buttons are linked to the history list, they can navigate to pages that were previously visited in a different window than the current one. This has implications for users that browse in different windows, carrying out separate tasks in each. For example, the user may be checking their stock prices in one window while reading a news article in another. Often users have multiple windows open so that they can read one page while waiting for a slow loading page to load in another. With Unified History, Back/Forward work with all pages in history. Thus, clicking Back could open pages that were viewed in the other windows, possibly causing confusion as pages from different sites would be intertwined.

3.3 Addressing revisitation issues

Having discussed the basic features of this system, I will now explain why these features were chosen and how they address the critical points made in the previous chapter.

3.3.1 There is a lack of integration between mechanisms

This problem described the cognitive and physical burden the user faces in dealing with separate systems for the Back/Forward buttons, bookmarks and history. Unified History integrates these mechanisms into a single unit, agreeing on a single organization scheme – the *recency ordered list*. To accomplish this, the three systems are modified in the following way. First, Back and Forward use Greenberg and Cockburn's "Recency with Temporal Ordering Enhancement" model. Second, the history list is sorted by recency. Third, bookmark functionality is incorporated into the history listing via explicit Dog Ears and implicit importance marking through visit count.

The combined result of these changes is a consistent interface between systems. Back/Forward and the history list are now intrinsically linked, as the history list shows the user exactly where Back and Forward lead.

The bookmarks are no longer separate entities, but are just specially marked items in the history list. Thus, the user no longer has to approach history and bookmark items differently. The user has a consistent process for finding and accessing pages, regardless of whether they are bookmark items or regular history items. There is no need to learn and use a separate mechanism for each.

3.3.2 There are identified problems with stack-based Back

Last chapter I discussed how users are confused about which pages can and cannot be visited with these buttons. To remedy this, Greenberg and Cockburn (1999) developed a new Back button behaviour termed, "Recency with Temporal Ordering Enhancement". This behaviour guarantees that all visited pages in history can eventually be reached by clicking back. This scheme more closely matches the way that 10 out of 11 users 'thought' Back and Forward work in the standard browsers (Cockburn and Jones, 1996, and Greenberg, Ho, Kaasten, 2000).

Of course, this change also has a trade-off. Namely, more clicks are required to move to pages up a hierarchy that would more be quickly reached in the stack model. This issue is discussed in detail in Section 2.2.2. Its benefit is that previously pruned pages are now reachable.

To determine how users find the difference between the stack and recency-based Back behaviours, we performed an experiment. There were two main findings. First, when users were asked to predict which page Back will lead to, the stack-based behaviour was easier to predict. Second, when users were not asked to predict, but merely to perform web tasks with each behaviour, there was no clear preference between the two systems. This suggests that users can adjust to a recency-based model, assuming there would be benefit to doing so. The entire report from this experiment is included as Appendix 2.

3.3.3 Pages are difficult to recognize in the history list

It can be difficult to recognize a page by only its title or URL. Thumbnail images overcome some of these problems because they are direct representations of the page as the user saw it unlike abstractions like the title or URL. Where many systems, such as Microsoft's IE use generic 'Internet' icons for items in the history list, Unified History uses high-quality thumbnail images. The size of the icon is 32 x 32 pixels. Of course, small thumbnails such as these can be difficult to recognize and especially difficult to discriminate between similar looking pages. For this reason, the system immediately displays a zoomed thumbnail, at 135 x 135 pixels as the user moves the mouse cursor over an item.

The choices for the thumbnail sizes are not arbitrary. Throughout the Windows operating system, 32 x 32 is the size for 'large' icons. While I could have chosen any size, there is an important trade off between displaying large, detailed icons, and fitting many items in a singe screen to prevent scrolling.

In order to make this decision, I ran an extensive user study to determine the size of thumbnail needed for a person to recognize a) the general web site a page was taken from, and b) the content particular to that page. This study is described in detail in Chapter 4.

As will be seen, the size needed to attain 60% recognition of the general web site is 96 pixels. At this size, a user with a standard window size of 600 pixels in height would only see a maximum of 6 items. The actual number would be closer to 4 when you consider the screen height needed for the other interface components and windows. Clearly it is not possible to accommodate this size without forcing the user to do an exorbitant amount of scrolling to find items – something many users loath immensely.

Our study found that at size 32x32, people can recognize roughly 13% of previously visited web sites and 5% of exact pages (see Section 4.3.1). However, the study tested user's recognition of thumbnails with no other aids. In Unified History's history list, the thumbnail is paired with the page's title. We believe that the two cues combined are certainly more useful than title or thumbnail alone. Thus, for the small sizes, I feel that the standard 32 x 32 serves is a useful balance between giving clues about the look of a page and allowing many items to be displayed in the list. Continuing the above example, we can fit 18 icons in single column, again assuming we can allocate all of this space.

For the zoom up view, it is important that the view facilitate precise identification of a page. This mechanism is important for helping the user determine whether the particular page is actually the desired one. For example, this feature is useful for scanning several pages that look similar and have the same title.

The user study found that at an average size of 135 x 135 pixels, people have better than 80% accuracy for recognizing the web site and above 60% accuracy for identifying the exact page. It should be noted that this study found users were significantly better at identifying the page by thumbnail than by title or URL. Thus, the zoomed up view at this size is provably a useful page representation.

3.3.4 Pages are difficult to find in the history list

Of course, even with the thumbnail enhancements, searching for an item out of a list of many items is certainly a difficult proposition. Thus, the list must have sorting and searching capabilities.

For sorting the list, there are many possibilities – date, title, URL – as I showed in the last chapter. I have chosen to sort the list by recency for several reasons.

First, it is important that it is easy to find the most likely candidates to be revisited. Tauscher and Greenberg's study (1997) found that recently viewed pages made up the largest proportion of revisits. They found that while revisiting accounts for 58% of navigation, revisiting to one of the previous 10 pages accounts for 43% of navigation. Thus, there is a strong probability that the user will revisit one of these past ten pages.

While access to recently viewed pages is important, Tauscher and Greenberg (1997) state that users still need access to more distant pages. They found that the remaining 15% are revisits to pages outside of the previous 10 pages. They argue that support for finding these pages is also important, as these pages may be extremely difficult to find by clicking through hyperlinks.

It is unreasonable to assume that a user is willing to search for an item by scanning throughout the entire list. The solution I decided on is based on Ben Shneiderman's Dynamic Query (Shneiderman, B., Williamson, C. and Ahlberg, 1992) as described in Section 3.1.6. This system is based on using search filters. I use three such filters – domain, title and visit count. Filters, unlike searches, present the results in the same area as the regular listing – there are no additional windows to open or close. The other advantage of filters is that they are incremental. For example, with the title filter, as the user types letters there is immediate feedback showing how many entries remain on the list. This is much less work than having the user type an entire phrase only to have it either unsuccessful or return a large number of entries to scan through.

3.3.5 There are identified problems with bookmarks

Standard systems require the user to bookmark the page as they encounter it. This is difficult in practice, as a page's importance is often recognized after the fact. This system incorporates the bookmark features inside the history list. Thus, making a bookmark in hindsight is a trivial process, provided the item can be found in the list. The user does not have to visit a page to Dog Ear it.

The other problem with bookmarks is the large amount of work it takes to manage a bookmark collection effectively. Recall that users put off organization. By incorporating the bookmarks into the listing the system automatically organizes them by what has been found to be the best predictor of revisiting – recency (Tauscher and Greenberg, 1997). Thus, there is a strong chance that the user will be interested in returning to that newly marked page, rather than a page marked a long time ago. Also recall that users rarely clean out bookmarks that are no longer needed, or no longer exist (Cockburn and McKenzie, 2000). Again, by incorporating them into the history list, the older items do not interfere with the newer, more relevant items. If the user does not visit a bookmark page for a long period, it migrates to the bottom of the list, eventually to scroll out of view.

3.3.6 Alternate systems have high costs of screen space

While there are many alternate systems that effectively mitigate some of these usability issues, they usually have a high cost for the user by either consuming large amounts of screen space, or requiring several actions to uncover and utilize their functionality.

This system uses a conservative amount of screen space, taking up roughly one quarter of the browser window width. Thus, it takes up no more room than the standard Internet Explorer history facility. Further, by integrating it directly inside Internet Explorer, the process of uncovering or hiding the history list requires just one toolbar button press.

3.4 Conclusion

The following table summarizes how Unified History addresses each of the usability issues described in the previous chapter. I am not proposing that Unified History is the ideal solution to the usability issues of today's revisitation facilities. Rather, Unified History demonstrates the merits of having a consistent theme for Back, bookmarks and history.

Table 3-1. Problems addressed by Unified History.

	Problem	Addressing Feature
1.	Lack of integration between revisitation facilities.	Back/Forward, bookmarks and history all sorted by recency and represented in the same listing.
2.	Problems with stack-based Back/Forward.	Recency based back/forward buttons. Buttons are integrated with the history list for viewing the ordering of pages.
3.	Difficult to recognize items in the history list.	Thumbnail images provide a reliable cue, especially when the page is badly titled or has an unrecognizable URL.
4.	Difficult to find items in the history list.	History list is recency ordered, as these items are most likely to be revisited. Filters for domain, title and visit count allow searching for items without spawning new windows or new representations.
5.	Problems with collecting/managing bookmarks.	Bookmarks integrated into history list, in recency-order, so that old, unused bookmarks migrate to bottom of list. The title and URL filter also make it easier to find a particular bookmark.
6.	Alterative systems have high cost of screen space.	History list uses no more space than IE's standard facilities.

4. Comparing Titles, URLs, Thumbnails

In chapter 2, I described how users find it difficult to recognize pages in the history list (problem 3). This is due to how the representation of pages– titles and URLs– often differs from what the user saw on the actual page.

In order to overcome this problem, a history needs to display pages in a way so users can recognize them. Thumbnail images seem like a good idea, as they are direct representation of what the user actually saw. In Chapter 2, I described how some research systems use thumbnail images to represent pages and in Chapter 3 we saw how they form an important part of Unified History.

However, I feel it is important in HCI research to validate ideas with real data, as other major components of Unified History have studies behind them: the advantages of recency (Tauscher and Greenberg, 1997), the merits of recency-based back (Cockburn and Greenberg, 1999; Greenberg, Ho, Kaasten, 2000), the utility of dynamic queries for rapid search (Shneiderman, B., Williamson, C. and Ahlberg, 1992). What is missing is whether users actually do recognize web pages better by thumbnail than by title or URL. Czerwinski, van Dantzich, Robertson and Hoffman (1999) performed a study where they compared how well users could retrieve web pages using a spatially organized history system with thumbnails versus the same system with the thumbnails removed. They found that the users were initially slower at retrieving pages without the thumbnails, but this difference diminished as users relied increasingly on the spatial location. While this finding confirms the value of spatial location for memory, it does not compare the value of thumbnails versus title or URL alone, as it focussed on the spatial location variable. To compare strictly thumbnails, titles and URLs, I devised and carried out an experiment[4].

This chapter begins with four research questions that frame the experiment. Then I describe the experimental design and process. Finally, I discuss the results and their implications to history list design.

[4] Chris Edwards assisted in designing, carrying out, and results analysis for this experiment.

4.1 Research Questions

This study investigated how well people recognize pages they have previously seen when shown representations of these pages as titles, URLs and thumbnails at various sizes. The study frames the following four research goals.

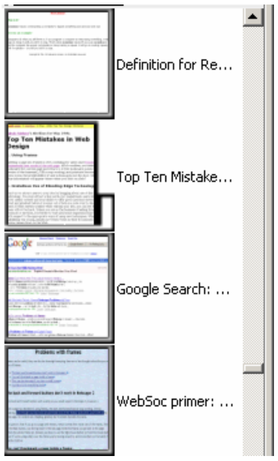
4.1.1 Thumbnail Size

Research Question 1: at what size thresholds do thumbnails have a reasonable chance of being recognized?

Size is obviously an important factor for thumbnail images[5]. The larger the thumbnail, the more it will resemble the page the user actually saw, and the more likely the user will recognize it.

However, there is a trade off between thumbnail size and the number of thumbnails that can be displayed in a linear history list. If we use too large a thumbnail size, the page may be recognizable but the user will only be able to see a few items on the list at a time (Figure 4-1a). Finding off-screen items requires scrolling, which is tedious. If the thumbnails are too small, the user may be able to see many thumbnails at a glance, but these thumbnails may not be recognizable and thus they would not provide much help (Figure 4-1b). The research question is a search for a reasonable trade off: at what size thresholds do thumbnails have a reasonable chance of being recognized? To be useful, we need to find a thumbnail size that balances its recognition with space demands.

[5] On most computer displays, "size" is a function of screen resolution (pixels per inch) and the number of pixels in the image. In this chapter, size refers to only the number of pixels in the image. In the future, we expect screen resolution advancements will make the number of pixels a less useful measurement compared to physical (centimetres/inches).

a. 

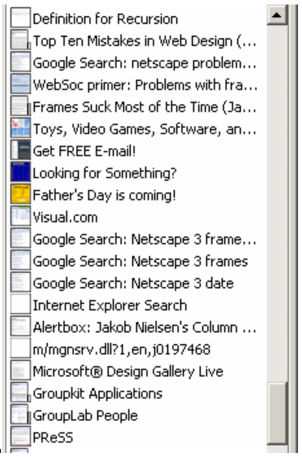
b. 
Definition for Recursion
Top Ten Mistakes in Web Design (...
Google Search: netscape problem...
WebSoc primer: Problems with fra...
Frames Suck Most of the Time (Ja...
Toys, Video Games, Software, an...
Get FREE E-mail!
Looking for Something?
Father's Day is coming!
Visual.com
Google Search: Netscape 3 frame...
Google Search: Netscape 3 frames
Google Search: Netscape 3 date
Internet Explorer Search
Alertbox: Jakob Nielsen's Column ...
m/mgnsrv.dll?1,en,j0197468
Microsoft® Design Gallery Live
Groupkit Applications
GroupLab People
PReSS

Figure 4-1. Thumbnail size determines how many items are visible in the history list.

4.1.2 Title Size

Research Question 2: what is the size threshold per truncation method that provides a reasonable chance that the page is recognized by title?

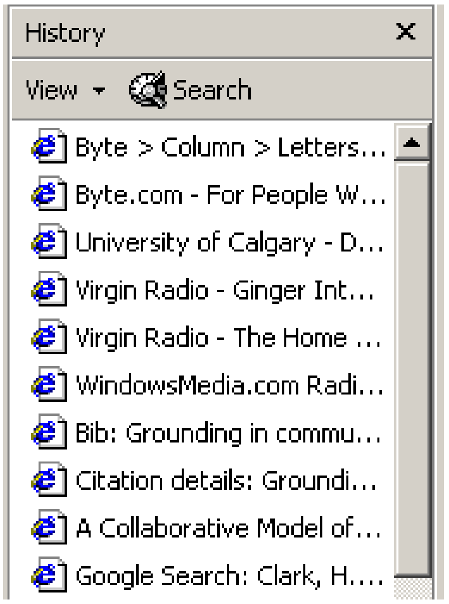

History
View
Search
Byte > Column > Letters...
Byte.com - For People W...
University of Calgary - D...
Virgin Radio - Ginger Int...
Virgin Radio - The Home ...
WindowsMedia.com Radi...
Bib: Grounding in commu...
Citation details: Groundi...
A Collaborative Model of...
Google Search: Clark, H....

Figure 4-2. IE's History list truncates titles.

There is a similar trade off with title size. In this context, size refers to how much of the text is visible to the user. The problem is that space-conservative history lists cannot fit the entire title within the narrow column. Instead, they present a shortened form of the title. For example, Figure 4-2 illustrates Internet Explorer's history bar, designed to occupy a conservative amount of screen width. The titles that do not fit are truncated.

Internet Explorer truncates the titles by cutting off the right hand side of the text (Figure 4-2). There are, however, three different approaches to truncating: right, centre, and left. Table 1 illustrates each approach by example, where the title "University of Calgary – Computer Science Home Page" is truncated into 30 letters.

Table 4-1. Truncation methods used in the experiment.

Approach	Example Title (30 letters)
Right	University of Calgary -- Compu...
Centre	University of C...ience Home Page
Left	... -- Computer Science Home Page

This example title is a real one, taken from the home page of the University of Calgary web site. It illustrates how the title reads quite differently with each truncation method. Right truncation shows only the title's beginning, so one sees in Table 1 that the page is from the University of Calgary, and guesses that it has something to do with computers. Centre truncation shows only portions of the beginning and end, so that one sees that it is from a university beginning with the letter 'C', and that it is some kind of homepage. Finally, the left truncation shows only the ending, so one sees that it is a Computer Science homepage, but there is no indication that it is from a university.

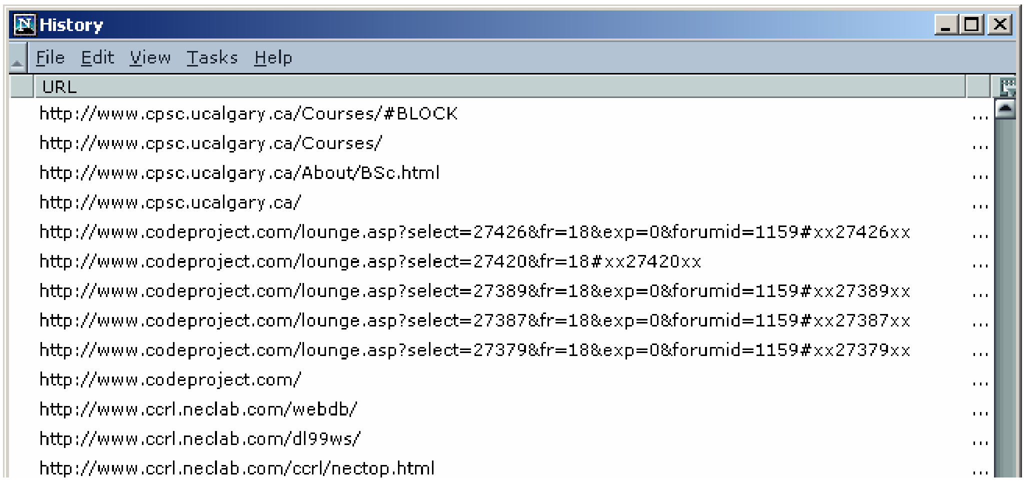
For titles to be useful, we need to determine how each title truncation method balances its recognition with size.

4.1.3 URL Size

Research Question 3: what is the size threshold per truncation method that provides a reasonable chance that the page is recognized by URL?

The same trade off also applies to URLs. However, the effects of truncation are likely more critical, as URLs often reach sizes far beyond titles. Figure 3 illustrates Netscape's history system, with the URL column widened so that we can see the various lengths of these particular URLs. Notice how the longest URLs seem to be computer generated, often containing long strings of numbers and codes that are largely incomprehensible.

Similar to titles, the left, centre, and right truncation methods can be applied to URLs. Table 2 illustrates how the example URL http://www.cpsc.ucalgary.ca/grouplab/software appears when we truncate it to 30 characters using the three different methods.



```
http://www.cpsc.ucalgary.ca/Courses/#BLOCK
http://www.cpsc.ucalgary.ca/Courses/
http://www.cpsc.ucalgary.ca/About/BSc.html
http://www.cpsc.ucalgary.ca/
http://www.codeproject.com/lounge.asp?select=27426&fr=18&exp=0&forumid=1159#xx27426xx
http://www.codeproject.com/lounge.asp?select=27420&fr=18#xx27420xx
http://www.codeproject.com/lounge.asp?select=27389&fr=18&exp=0&forumid=1159#xx27389xx
http://www.codeproject.com/lounge.asp?select=27387&fr=18&exp=0&forumid=1159#xx27387xx
http://www.codeproject.com/lounge.asp?select=27379&fr=18&exp=0&forumid=1159#xx27379xx
http://www.codeproject.com/
http://www.ccrl.neclab.com/webdb/
http://www.ccrl.neclab.com/dl99ws/
http://www.ccrl.neclab.com/ccrl/nectop.html
```

Figure 4-3. Netscape's (version 4.3) history list, with URLs of various lengths.

Table 4-2. Truncation methods applied to URLs.

Approach	Truncated Text
Right	http://www.cpsc.ucalgary.ca/gr...
Centre	http://www.cpsc...ouplab/software
Left	....ucalgary.ca/grouplab/software

Again, each method reveals different aspects about the page – that it is from the University of Calgary in Canada (right truncation), that it refers to software (centre truncation), and that it is the software portion of the Grouplab research group (left truncation).

4.1.4 Distribution of Title and URL Length on the Web

Research Question 4: what is the distribution of lengths for titles and URLs from pages typically found on the Web?

In Figures 4-1 through 4-3, we saw that both titles and URLs vary greatly in their size (i.e., string length). Some are quite short and fit easily in even a narrow column, while others are very long. If most titles/URLs are short then truncation is not that important. If they are long, we can expect much truncation. Either way, we need to investigate the distribution of titles and URLs to place answers from research questions 2 and 3 in context.

4.2 Method

While the goal is straight forward, constructing a procedure to answer the above research questions turned out to be a difficult challenge. The problems stemmed from the well-known conflict between designing a controlled and replicable study versus attaining results that apply to real world situations. My primary goal of this research is to find out what kinds of representations work well in a history list under real world web navigation. Thus, to achieve a balance between internal and external validity, I performed a pseudo-controlled study to best mimic real world situations.

4.2.1 Variables

The independent variables were the representation type shown to the subject (thumbnail, title, URL) and truncation method (right, centre, left) for titles and URLs. The dependent variables were two size thresholds at which the subject could identify the web site, and then the exact page from a thumbnail, as well as the accuracy of these identifications. The qualitative data were the subjects' descriptions about how they were able to identify the page, including their spoken comments.

4.2.2 Subjects

We recruited 20 paid participants, all computer science students, who were either 2nd year or higher in the undergraduate program or who were graduate students. The important factor is that all were practised Internet users.

Using computer scientists as subjects, we expected some performance differences when compared to the 'normal' population of Internet users. In particular, we expected this group would be more comfortable recognizing pages by URL addresses than novice users. Thus, this group provides a 'best-case scenario' for determining the effectiveness of URLs. If URLs turned out to be ineffective for this group, it is likely that URLs would be ineffective for other groups as well.

4.2.3 Stimuli

The goal of the study is to test subject's recognition of pages they had previously visited. This implies two phases in the study: a priming phase where the subject looks at a chosen set of web pages, and a test phase where the subject attempts to recognize selected pages from their different visual representations.

To do this as a controlled study, we should prime subjects with the same set of pages. However, finding a good set of candidate pages introduces several serious problems, most relating to how we could generalize our results to browser design.

a. **Artificiality of page interest**. Subjects may have no personal connection with the set of pages we give them. This could profoundly affect how well (or how poorly) they remember these pages. In real use, we expect people will attend to various pages quite differently, due to their immediate interest or page appeal.

b. **Artificiality of learning**. The way we ask subjects to 'learn' pages could also profoundly affect how well they are remembered. We could insist that they read each page, or have them search page contents for information, or merely present pages for a time duration. In real use, we expect people to 'learn' pages differently, as a function of their interest, how much they read them, what they are looking for, etc.

c. **Page Composition.** Pages on the web are remarkably inconsistent. In terms of visuals, they vary greatly in their typographic structure (use of proximity, white space, fonts, contrast), and their graphical elements (image type, quantity, size and noise such as advertisements). Similarly, pages vary greatly in how titles and URLs are composed (see Section 2). There are virtually no statistics that describe common page attributes. If we 'make up' our own pages, the ability of people to recognize them may have little bearing on how they recognize perhaps quite different pages on the web.

Consequently, we decided to use the actual pages the subject had viewed during normal browsing activity as stimuli, making the process a pseudo-controlled experiment. We had each subject submit his or her history record to us. From this list, we randomly selected 30 pages for the experiment. By using these pages, this experiment looked at how different representations worked for real users and their real pages.

Of course, using pages from the subject's actual history record has some potential ethical implications. Web pages can contain very private information, some that many people would not feel comfortable showing someone they just met. We reduced risk by assuming that all password protected pages (e.g. online banking) are sensitive and then making sure that these pages were excluded from our list of 30. Another problem could be the risk of embarrassment, if people (say) visited 'undesirable' sites such as pornography sites. While this problem was largely avoided because most subjects' history records came from their web usage on public University labs, we informed all potential subjects of this risk (see Appendix 2). Before signing up for the experiment, we told all subjects how we would protect their private information and that the pages selected for the experiment were accessible to only the two lead experimenters.

4.2.3 Materials

The materials comprised of a computer, standard web browser software, and custom experiment software.

- *Computer*. We performed the experiments on a high-end computer running Windows 2000 using a 17-inch monitor running a resolution of 1152x864 with 32-bit colour.
- *Browsers*. All subjects were either Internet Explorer or Netscape users before the study, for we could only extract the history record from those two browsers. At the time of the study, the latest release was Internet Explorer 5 and Netscape 6. We also used Internet Explorer to display pages to the subject during the verification phase (see 4.2.4).
- *History Extraction Software*. This custom software generated a text file consisting of the pages that the subject had visited within Internet Explorer. Entries included the pages' URL, title and date last visited. The subjects who used Netscape did not need to use this software, as they could use Netscape's history list 'save-as' option.

- *Stimuli Preparation Software*. This custom software read in the history file, displayed each page, let the experimenters select 30 pages and generated a high definition thumbnail image of a selected page[6]. It later asked subjects to verify the accuracy of their responses and judge the quality of the particular page representation.
- *Stimuli Presentation and Verification Software*. This custom software presented the stimuli to the subjects and recorded the subjects' responses.

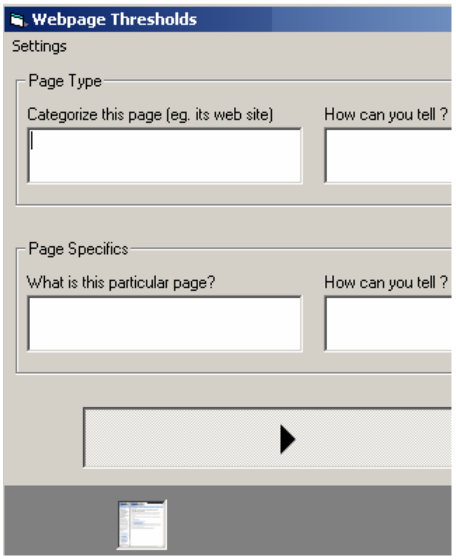
4.2.4 Procedure

Step 1 : Stimuli Preparation. First, the subject submitted their history record to us. For Netscape users, the history list has a simple option to save the items to a file. For Internet Explorer, we gave them the custom *History Extraction* software. From this list, we selected 30 pages. We manually selected these pages in order to filter out problematic ones: no title, frames[7], password protected, or broken/slow-loading pages. Also, we excluded pages if several had already been selected from that particular web site. The period of history varied for each subject. For most subjects, it included the pages visited within the last 3 weeks. Using the *Stimuli Preparation* software, we then captured the thumbnail images that would be needed for the test.

Step 2 : Stimuli Presentation. The experiment procedure took place about 1-3 days after the subject submitted the history file. The subject first read and signed the consent form. We then read the pre-experiment script (Appendix 2). Using our custom *Stimuli Presentation* software, the basic procedure for each trial was to show a particular page representation at an extremely small, probably unrecognizable size, and to gradually increase the representation size until the subject could just recognize and say what web site the page came from. The subject would then continue until he or she could identify the specific page.

[6] The method for capturing the thumbnail is described in Section 5.4.

[7] If a page utilizes frames, the history will contain entries for each individual frame. This issue is discussed in Section 5.2.3.



Webpage Thresholds

Settings

Page Type

Categorize this page (eg. its web site)

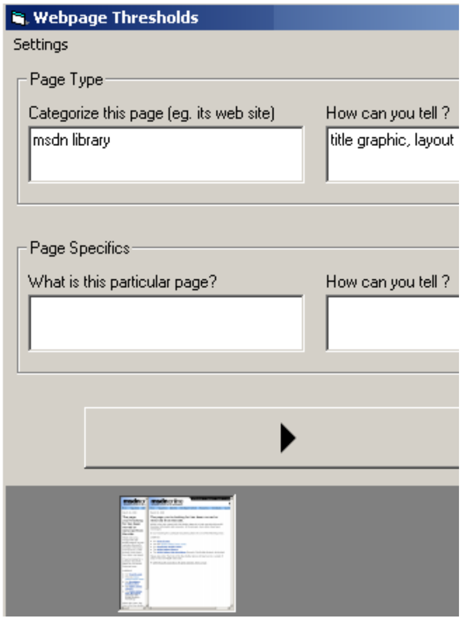
How can you tell ?

Page Specifics

What is this particular page?

How can you tell ?

Figure 4-4a. Thumbnail appears at 16x16 pixels.



Webpage Thresholds

Settings

Page Type

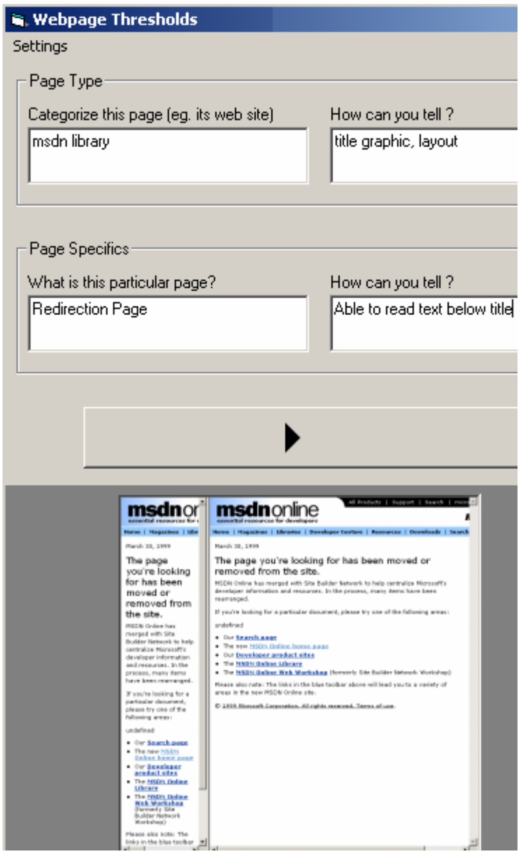
Categorize this page (eg. its web site)	How can you tell ?
msdn library	title graphic, layout

Page Specifics

What is this particular page?

How can you tell ?

Figure 4-4b. Subject stops the growing at 36x36 pixels, identifies web site.



Webpage Thresholds

Settings

Page Type

Categorize this page (eg. its web site)	How can you tell ?
msdn library	title graphic, layout

Page Specifics

What is this particular page?	How can you tell ?
Redirection Page	Able to read text below title

Figure 4-4c. Subject identifies exact page at 108 pixels.

For thumbnails, the image first appeared scaled down to 16x16 pixels. For titles or URLs, the initial string size is two letters. Depending on the truncation method used, this meant the subject saw the first two letters, the first and last letter, or the last two letters.

Figure 4-4 illustrates a typical thumbnail trial sequence. In Figure 4-4a, the subject first sees a thumbnail image, at only 16x16 pixels size. The thumbnail then automatically begins to grow larger, incrementing at a rate of 16 pixels every 3 seconds. The subject watches the thumbnail as it increases, until he just recognizes what web site it came from. At this point, he clicks on the large 'play/pause' button in the middle of the screen to stop the thumbnail from growing. The subject then types in the web site's description in the top left field (labelled 'page type'), and how he recognized it in the top right field (labelled 'how can you tell?'). For example, in 4-4b we see that he has paused the thumbnail at size 36, typed in the web site name "msdn library" and that he recognized it by its title graphics and layout. The subject is still is not sure what the exact content is on this page, so he presses the 'play' button to have the thumbnail continue growing. Finally, he recognizes the page at size 108, clicks 'Pause' and fills in the exact page name and how he recognized it (Figure 4-4c). He then clicks the 'next' button (not shown) to proceed to the next trial web page to be tested.

The sequence for the textual representations worked exactly the same way. The title or URL was initially truncated to display only two letters using one of the left, centre or right methods. The *Stimuli Presentation* software revealed the text at a rate of two letters every 3 seconds. Figure 4-5 shows an example, where a URL is being revealed using the truncate-right method.

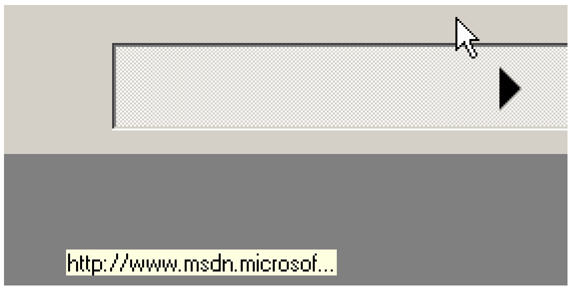
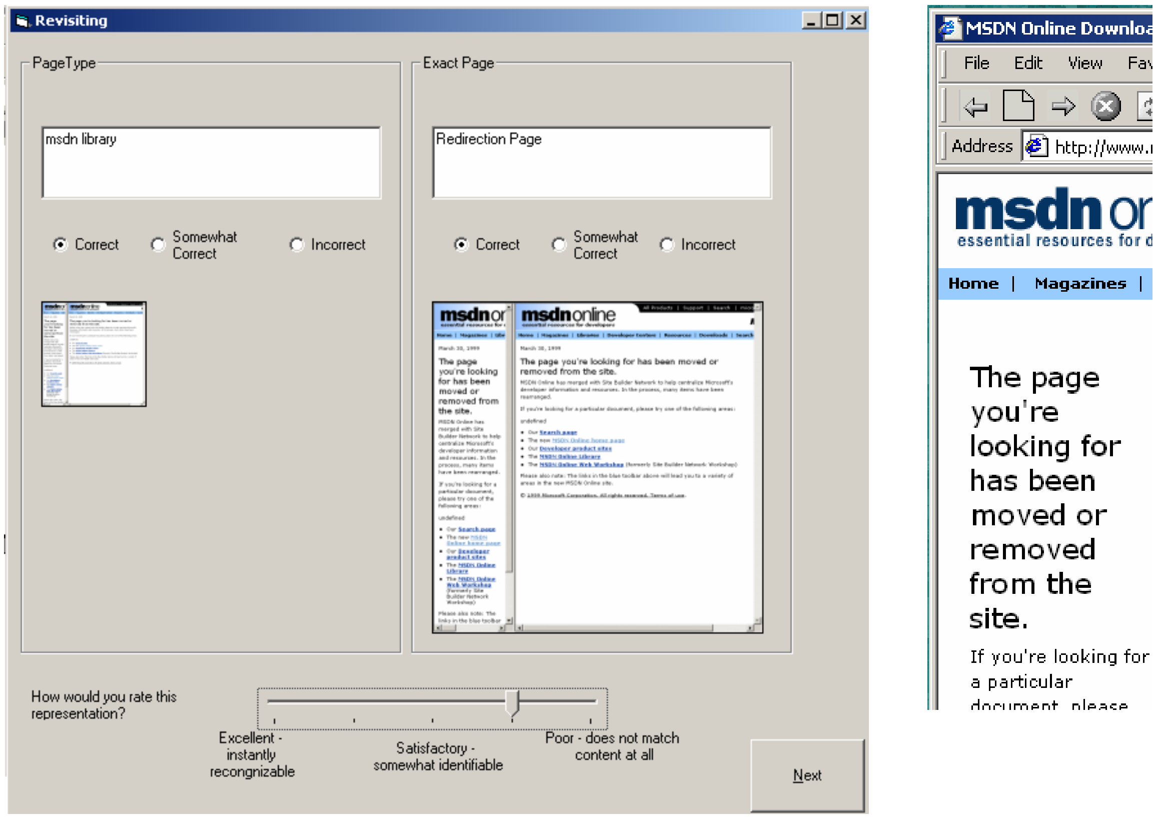


Figure 4-5. Subject being shown a URL, using the truncate-right method

The subjects saw 30 pages; thus they had 30 trials. Each trial used only a single presentation, alternated in the following sequence: thumbnail, title, URL. The titles and URLs alternated themselves between the right, centre and left truncation methods. This pattern balanced the three types of representations: 10 thumbnails, 10 titles and 10 URLs. The truncation methods were not balanced. For titles, there were 4 truncate-right, 3 truncate-centre and 3 truncate-left. For URLs, there were 3 truncate-right, 4 truncate-centre, and 3 truncate-left. In hindsight, we regret not balancing the truncation methods, but do not expect that this had serious effect on the data.

Step 3 Stimuli verification. Once all 30 trials were completed, the verification process began. During this phase, subjects went through their entries to see if they had correctly identified the pages. We did this to verify that the threshold measurements (thumbnail, title, or URL size) occurred at the point where the site was correctly recognized.


Revisiting

PageType: msdn library

[✓ Correct] Somewhat Correct Incorrect

Exact Page: Redirection Page

[✓ Correct] Somewhat Correct Incorrect

How would you rate this representation?

Excellent - instantly recongnizable | Satisfactory - somewhat identifiable | Poor - does not match content at all

Next

MSDN Online Downloa

File Edit View Fav

Address http://www.

msdn or

essential resources for d

Home | Magazines |

The page you're looking for has been moved or removed from the site.

If you're looking for a particular document, please

Figure 4-6. Evaluating the subject's answers.

For each page, we showed the subject that page in the regular web browser along with the form in Figure 4-6, displayed by the *Stimuli Presentation and Verification* software. This form displayed that page in the representation they saw at the two sizes he/she indicated as just being able to recognize the web site (left side of Figure 4-6) and exact page (right side of Figure 4-6). The form also asked several questions. First, the subject indicated if his/her answer for the web site and the exact page was 1) Correct, 2) Somewhat correct, or 3) Incorrect (figure 6 top). These categories mapped into: 1) the site or page is exactly what the subject had in mind, 2) it is very close to the site or page in mind, and 3) not the correct site or page at all. We used this system rather than just the simple correct/incorrect because in practice, a history system is useful if it provides approximate, if not precise, access to a given page. There is value in getting close to the desired page. The subject had the final say on the correctness of the response, as they were the best judge as to whether the page matched the page they thought of in their head.

The subjects used the same form to rate how well the representation 'captures' the page (Figure 4-6 bottom). For titles, this was a rating of the complete title, not just the portion he or she saw before answering. Likewise for URL, the subject was asked to rate the complete, non-truncated URL. For the thumbnail, the question did not refer to a specific size, but rather the concept – does the 'look' of the full sized page, as seen in the browser, give a good indication about its content? We asked this question in order to judge the representations from a different perspective. There could be situations where the subject did not recognize the page simply because he or she only briefly glanced at it, which cannot be blamed on the representation. This question allowed the subject to compare the chosen representation with the actual web page to rate how well they match.

These ratings were done on a Likert scale, ranging from 1) Instantly recognizable to 5) Does not match content at all. The subject answered this by moving a slider, shown at the bottom of Figure 4-6.

Additionally, the subjects were told to base their answer on the representation's effectiveness for both web site and exact page identification. One argument combining these into a single value is that the page will only have one title, URL, or visual thumbnail (at multiple sizes) that the user will see to make both the web site and exact page identification.

4.3 Results and discussion

4.3.1 Thumbnail size

Research question 1 was: at what size thresholds do thumbnails have a reasonable chance of being recognizable?

Results. Figures 4-7a-b plot the threshold distribution where subjects where just able to identify the web site (4-7a) and the exact page (4-7b). The graphs also illustrate which of these pages were correctly, somewhat correctly or incorrectly identified. Figure 4-7c plots the data as a cumulative distribution, where each point is the running sum of all previous points. Only the correct and somewhat correct responses are included in the cumulative distribution, as we are only interested in the sizes where the thumbnail was useful for the subject.

Figures 4-7a and b locate the mean thumbnail size needed to identify the web site as 98^2 pixels (std. dev.=28) and to identify the exact page as 135^2 pixels (std.dev.= 29). While this mean gives us a ballpark threshold for recognizing pages by thumbnail, the remaining discussion will concentrate on the threshold distributions, as they provide more useful information.

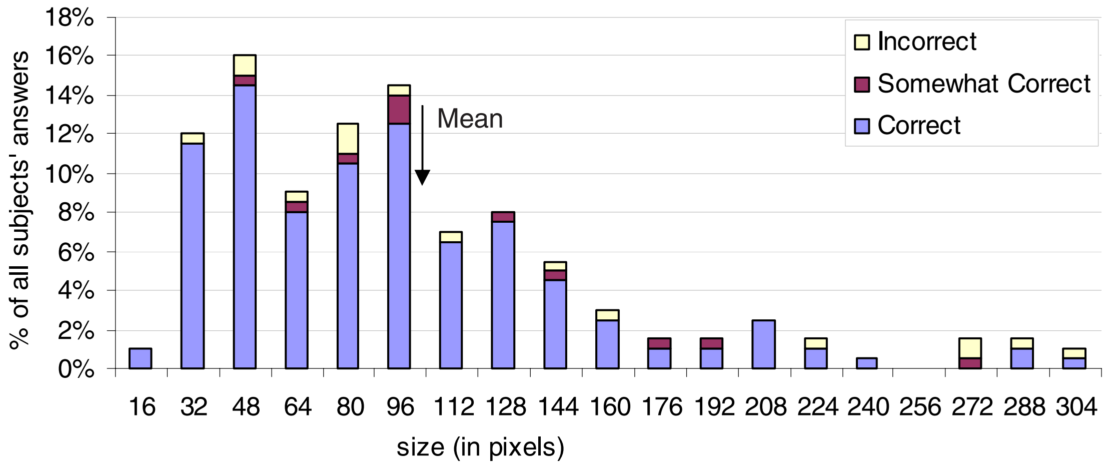


Figure 4-7a. Distribution of thumbnail size for identifying web site.

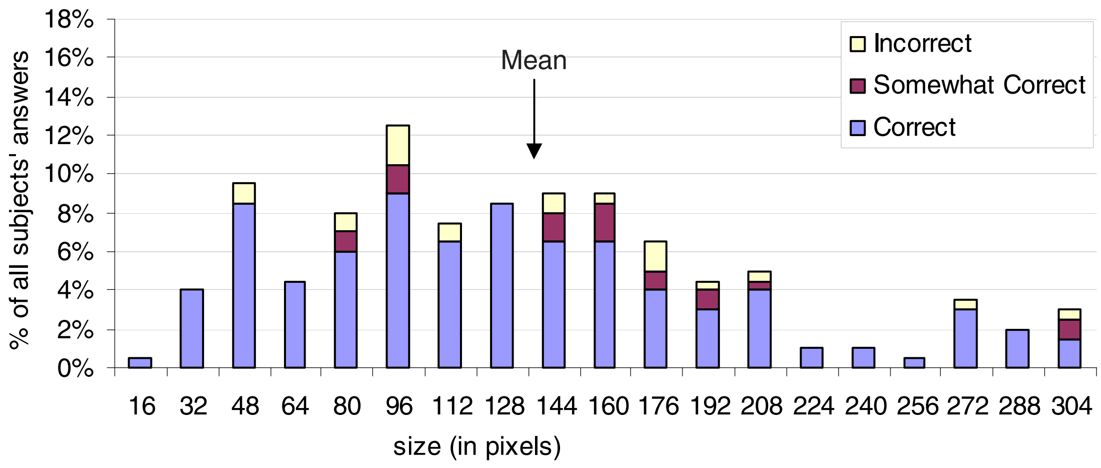


Figure 4-7b. Distribution of thumbnail size for identifying exact page.

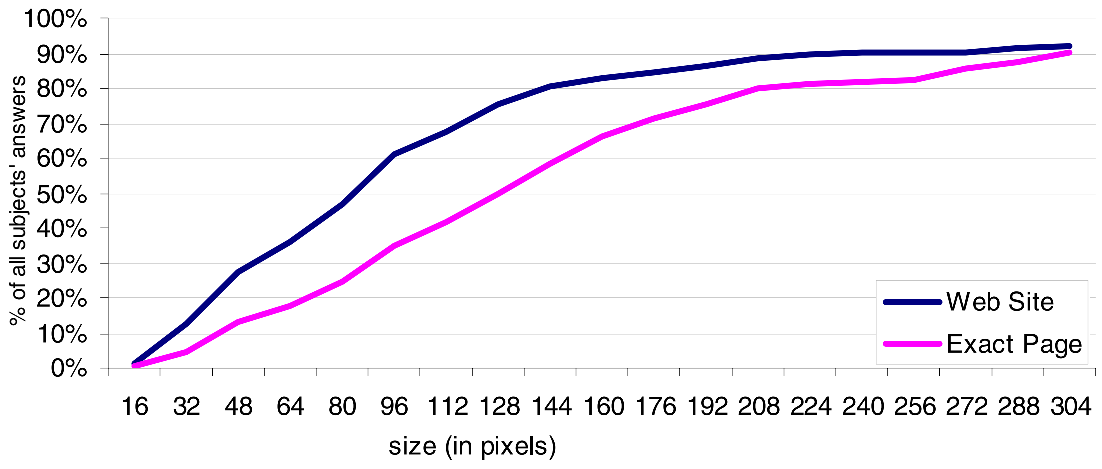


Figure 4-7c. Running sums of thumbnail sizes for identifying web site and exact page.

We see in the somewhat negatively skewed distribution of Figure 4-7a that web site recognition at thumbnail size 16^2 pixels barely totals 1%, which means that this is too small to be useful. However, thumbnails quickly gain value after that. For site identification, sizes between 32^2-96^2 pixels are quite useful (Figure 4-7a), where subjects had identified the web site for 60% of all thumbnails sized 96^2 pixels or less (Figure 4-7c). Increasing the size beyond 96^2 pixels still provides incremental benefit (80% recognition at 144^2 pixels or less), but recognition rates increase only marginally with size. We also see in the more normally distributed distribution of Figure 4-7b that thumbnails sized 32^2 pixels or less are not very useful for identifying the exact web page, with a recognition rate of only 5%. Increasing the size one increment to 48^2 pixels gives a fairly large jump (almost 10% at that size only). Thumbnail recognition then improves somewhat steadily after that, with near 60% recognition at 144^2 or less, and 80% at 208^2 pixels or less (Figure 4-7c). Recognition improves only marginally beyond that size.

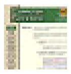
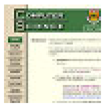
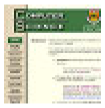
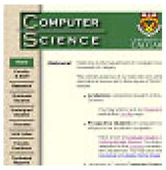
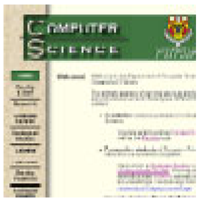
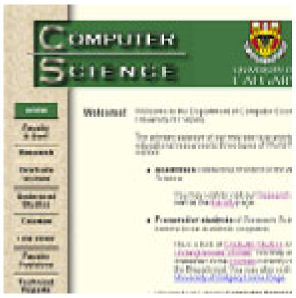
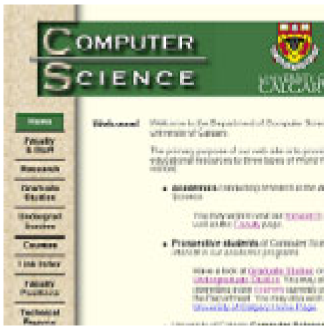
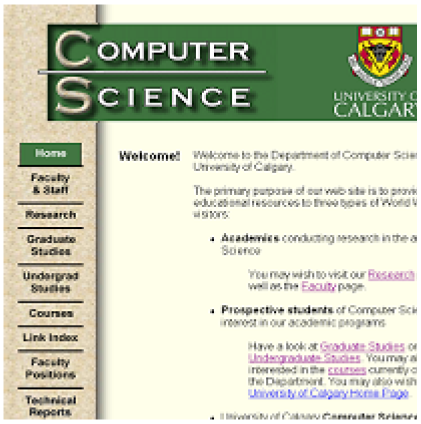
Discussion. The previous described data, and especially the cumulative distribution of Figure 4-7c gives us a cost-benefit guide for the effectiveness of thumbnail size. We assume that showing a person a thumbnail at only one particular size is equivalent to the cumulative effects of seeing the thumbnail at all of its smaller sizes. That is, a larger thumbnail will be at least as recognizable as all of its smaller versions. Thus, the cumulative distribution is more useful than the individual distributions, as it gives a recognition value for a given thumbnail size. For that reason, it is the basis of our recommendations.

In order to make recommendations, we need to decide on benchmarks for recognition. While these benchmarks are arbitrary to a certain degree, they nonetheless allow us to make recommendations for thumbnail, title and URL size that are consistent with each other. We will set the benchmarks as 15%, 30%, 60% and 80% for minimum, low, medium, and high recognition levels respectively. Of course, developers can choose their own benchmarks, and find the recommended size by looking at Figure 7c.

If space is very tight, the minimum useful size for a thumbnail is 32^2 pixels for identifying web sites, and 48^2 pixels for identifying exact pages. If space demands are somewhat less stringent, low recognition (~30%) is achieved with 48^2 pixels for web sites and 80^2 pixels for exact pages. For medium recognition (60%), we need 96^2 pixels for web sites and 144^2 pixels for exact pages. Finally, for high recognition (80%), we need 160^2 pixels for web sites and 208^2 pixels for exact pages. There is little benefit gained from having thumbnails any larger than these sizes. These recommendations are summarised in Table 4-3, including an example thumbnail scaled to each of these sizes.

We should also add that these recommendations take into account subjects' accuracy at correctly identifying web sites and pages. First, the accuracy rate was very high (above 80% were identified correctly). Second, errors that did occur happened at all sizes, as illustrated in Figures 7a-b. Thus, attempts to recognize thumbnails at small sizes did not lead to more errors than attempts at larger sizes. Accuracy will be discussed further in Section 4.3.6.

Table 4-3. Thumbnail Size Recommendations

Benchmark	Size needed to identify web site	Size needed to identify exact page
Minimum (15%)	 32^2	 48^2
Low (30%)	 48^2	 80^2
Medium (60%)	 96^2	 144^2
High (80%)	 160^2	 208^2

4.3.2 Qualitative thumbnail results

Each time the subjects made a guess at the web site or exact page, they specified what the predominant features that influenced their guess[8]. These comments invariably dealt with the following visual attributes:

- *Colours* - background colours and font colours for the page;
- *Text-Related* - legible text from the title or secondary titles on the form. Very often, this title is graphical in nature and different from the page's html <title> tag;
- *Image-Related* – a distinctive image on the page;
- *Layout-Related* - the format and placement of the various elements on the page;

In order to analyze these comments, we categorized the subjects' answers into these attributes and counted how often they occurred. Often subjects mentioned more than one, such as "colours and layout'. Thus, a single thumbnail identification could count in both the 'colours' and 'layout' attributes. Figure 4-8 plots the total counts of these attributes for both web site and exact page identifications.

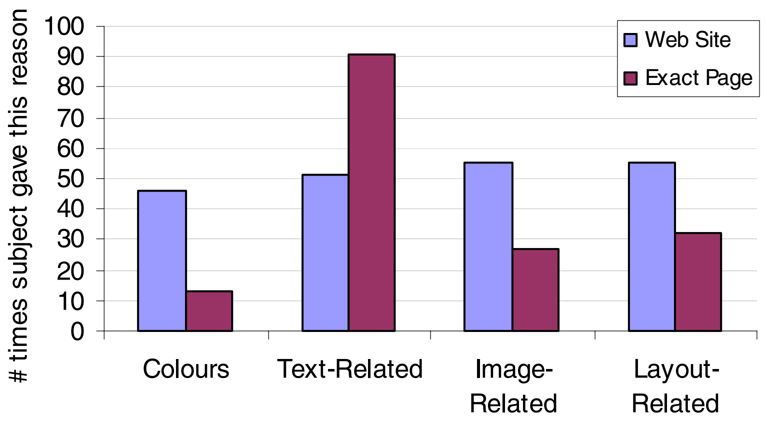


Figure 4-8. Predominant attributes in thumbnail that aided recognition.

[8] Subjects were not required to enter this information for every identification. Out of the 200 thumbnail trials, there were 185 comments for identifying the web site, and 172 comments for identifying the exact page.

The totals in Figure 8 indicate for identifying a web site, each attribute was equally likely to be responsible for being recognized. That is, subjects used all the attributes evenly to determine web site. However, the totals for identifying the exact page were quite different. Colours, layout, and images were of minor help for identifying the exact page. We see in Figure 4-8 that the subjects relied mostly on reading text inside the thumbnail. Of course, this implies that the thumbnails were large enough for the subjects to read the text. Thus, the size of the thumbnail is an important factor to revisit.

Figure 4-9a-b plot the distribution for the thumbnail stop sizes, similar to Figure 4-7. For each size, the graph illustrates how many times subjects mentioned each attribute.

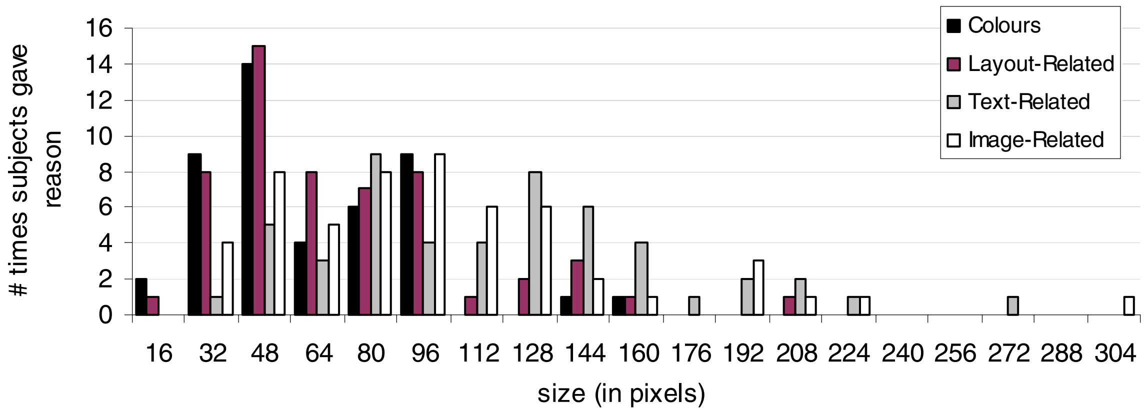


Figure 4-9a. Distributions of the reasons subjects were able to recognize web sites.

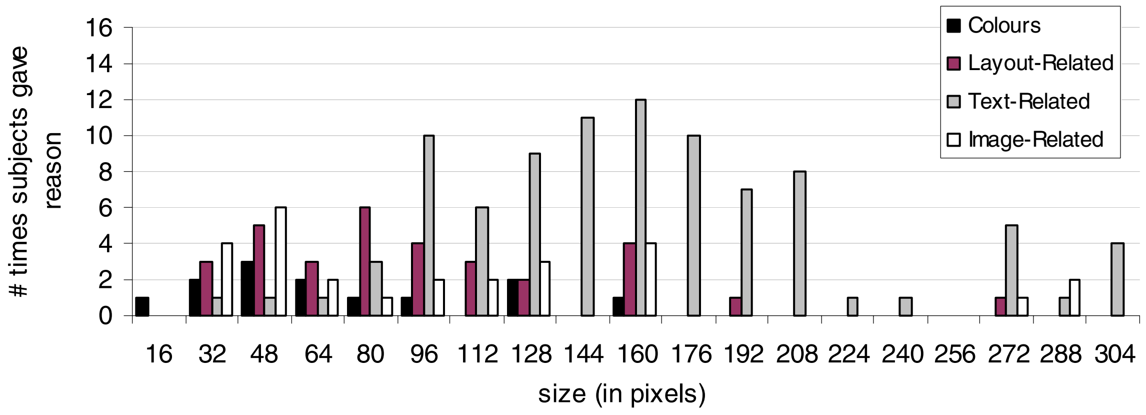


Figure 4-9b Distributions of the reasons subjects were able to recognize exact pages.

In Figure 4-9a, we see that when identifying web sites, the identifications that occurred early (less than 64 pixels), were primarily due to the colours or layout. If the subjects could not identify the site at a small size, they often waited for the thumbnail to grow above 100 pixels so that the text could become legible.

This trend is quite evident in the exact page identifications shown in Figure 4-9b. Nearly all of the ‘late’ identifications (when the thumbnail reached a size greater than 100 pixels) were based on reading text in the thumbnail.

Figures 4-10a-b plot these distributions as running sums.

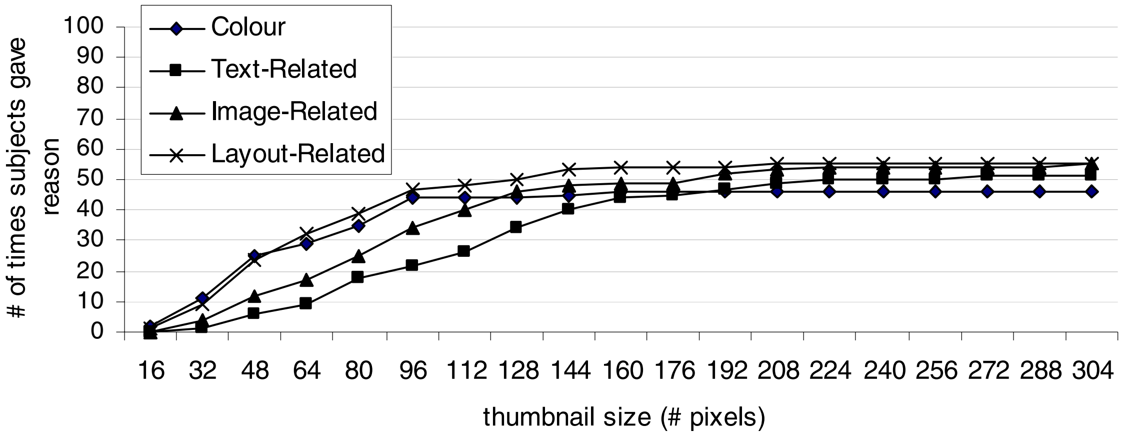


Figure 4-10 a. Running sum of reasons subjects were able to recognize web sites.

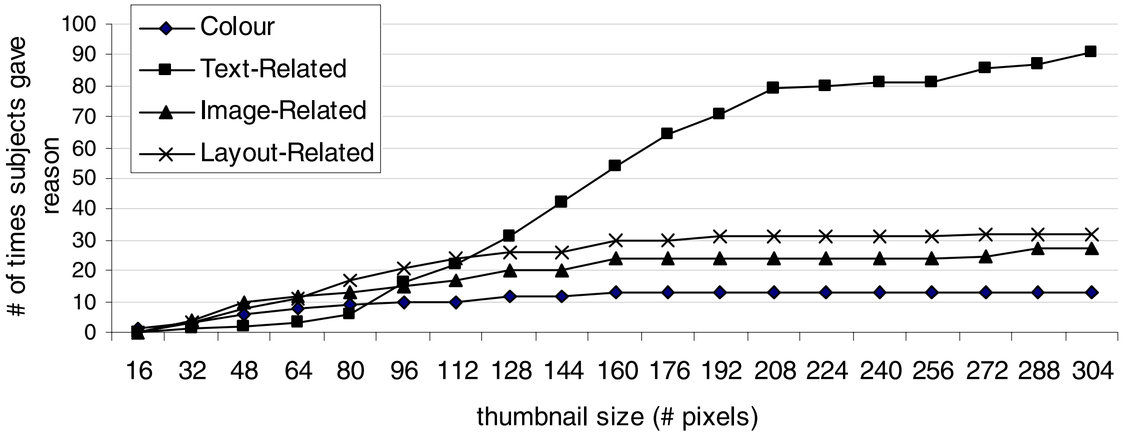


Figure 4-10 b. Running sum of reasons subjects were able to recognize exact pages.

In Figure 4-10a, we see again that subjects could identify small thumbnails by recognizing the colour and layout attributes. However, the flattening of the slope after 96 illustrates colour and layout have diminishing value. With the larger thumbnails, the use of text-related attributes increased.

Figure 4-10b shows a very different picture for identifying exact page. First, recall that only a modest number of exact page identifications took place with thumbnails smaller 96 pixels. Subjects needed larger thumbnails, and the vast majority of identifications were based on reading text-related cues, as shown by Figure 4-10 b.

Discussion. These results suggest what cues enable web site and exact page recognition. Very often, subjects could identify small thumbnails (less than 96 pixels) by the colour and layout rather than details. This is likely because many web sites have a distinctive 'look' that can be recognized in a small image icon.

For identifying the exact page, being able to read some of the page's text is clearly important. We could argue that using a thumbnail to display text defeats its purpose of using graphics, for instead we could simply display the text at a font size that is much easier to read than in a shrunken graphic. However, it is important to realize that the text that appears on the page, and therefore in the thumbnail, is often different from the page's technical title. For example, in Chapter 2 Figure 2-5 we saw that a page's technical title, "FACTS" did not appear anywhere on the page. This mismatch also occurs when the page's main title is a graphical banner, for the highly identifiable logo/banner cannot be expressed with regular text as well as it can through its image.

Finally, while text is clearly important, the surrounding colours, page layout and images likely provide both context and redundancy to make the page recognizable.

These results have implications for web site and page design. First, these results re-enforce the value of repeating colour/layout/images across pages, for pages become recognizable as coming from a particular site. Second, if thumbnails become an important interface feature then page designers should be encouraged to use large title/banner font sizes that are visible in small thumbnails.

4.3.3 Title size

Research question 2 was: at what size threshold per truncation method do titles have a reasonable chance of being recognized?

Results. First, we look at the mean sizes taken from when the subjects could identify pages using the various truncation methods. Figure 3-11 illustrates the means and standard deviations for each method. For identifying the web site, the subjects had the shortest mean size using the standard truncate-right algorithm (ANOVA $p \leq 0.05$, F = 5.10). With this truncation method, the subjects needed to see on average, the first 16 letters (standard deviation = 5.5) of the title before they attempted to identify the web site.

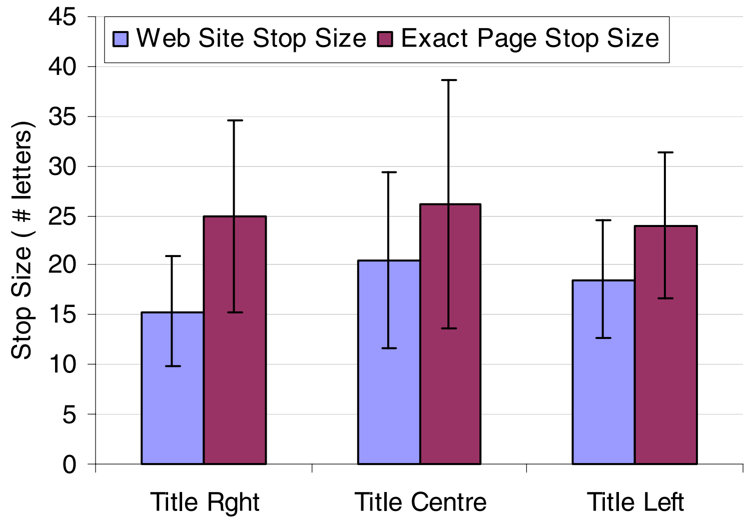


Figure 4-11. Subjects' mean stop sizes for identifying pages using title truncation methods.

For identifying the exact page, there was no significant difference between the different methods. All three methods required from 24-28 letters on average, with standard deviations of 10, 13, and 7 for the right, centre and left methods respectively. As before, these means only give ballpark thresholds, and thus we will look at the distributions that are far more informative.

Figures 4-12, 4-13 and 4-14 a-c illustrate the stop size distributions for each of the title truncation methods. The cumulative distributions only include the correct and somewhat correct responses in order to isolate the 'useful' sizes. Figures 4-15 a-b combines Figures 4-12c, 4-13c and 4-14c to help us visually compare the three truncation methods for identifying web site and exact page.

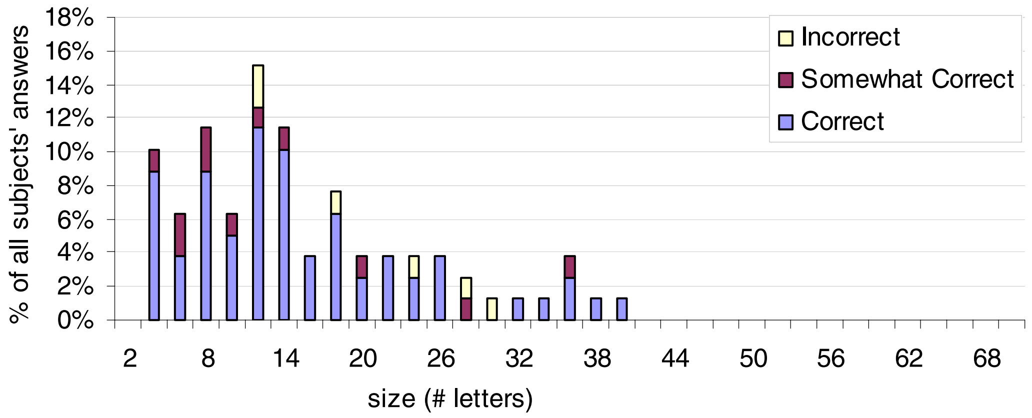


Figure 4-12 a. Distribution of title stop sizes using truncate-right method to identify web sites.

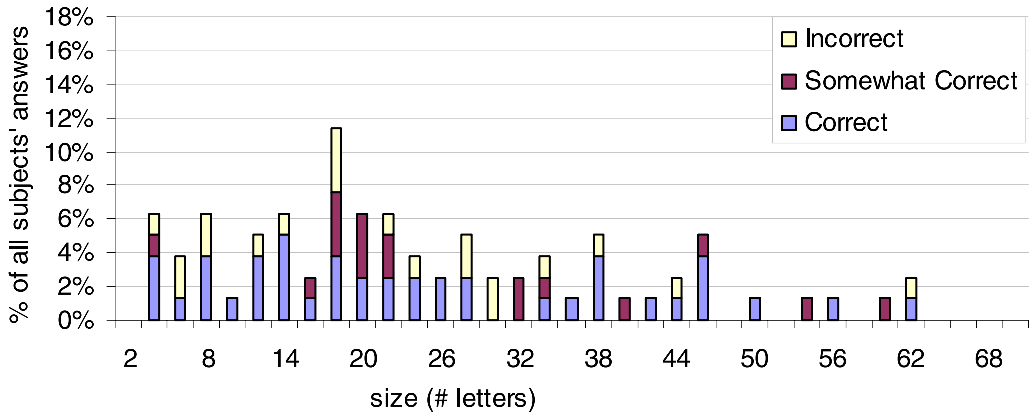


Figure 4-12 b. Distribution of title stop sizes using truncate-right method to identify exact pages.

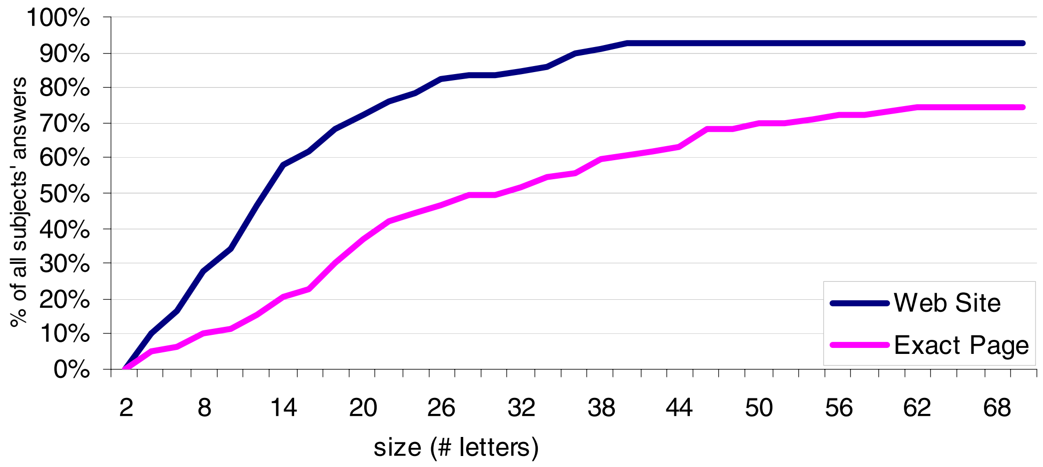


Figure 4-12 c. Running sums of title stop sizes using truncate-right method.

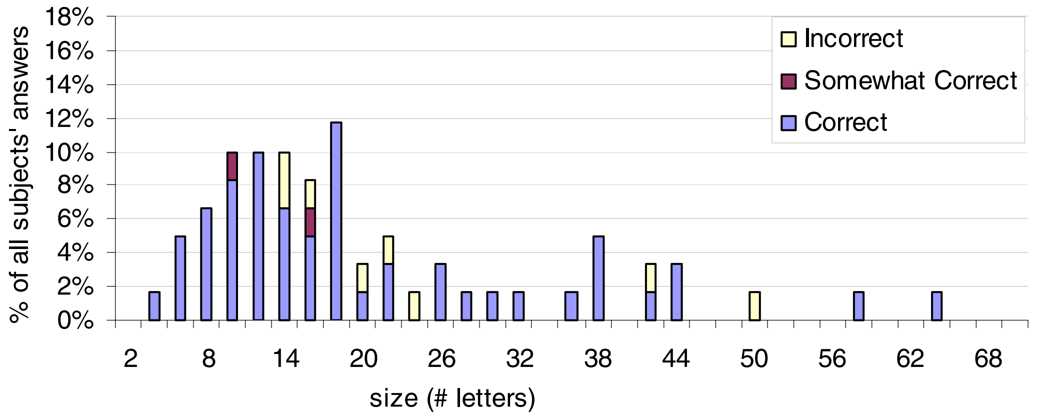


Figure 4-13 a. Distribution of title stop size s using truncate-centre method to identify web site.

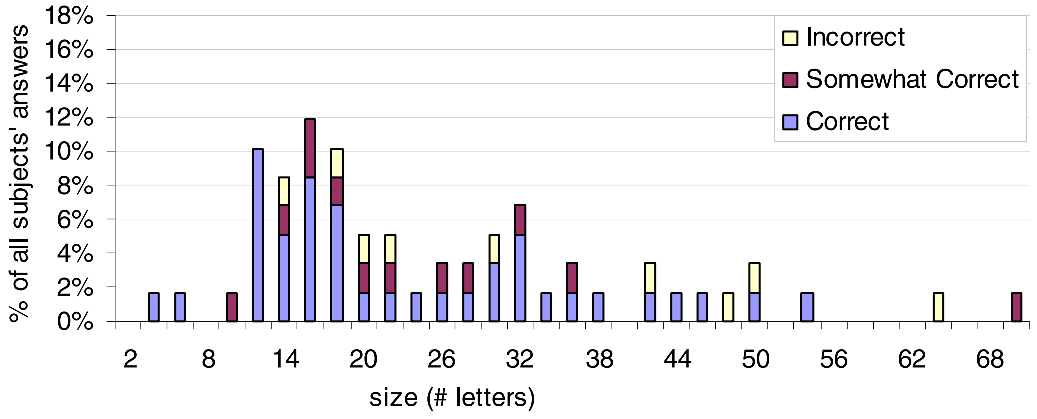


Figure 4-13 b. Distribution of title stop sizes using truncate-centre method to identify exact page.

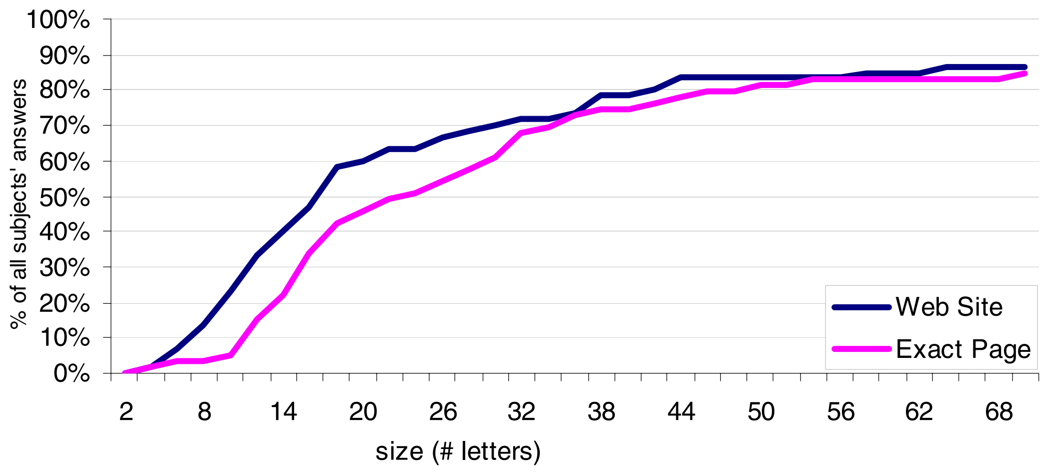


Figure 4-13 c. Running sums of title stop sizes using truncate-centre method.

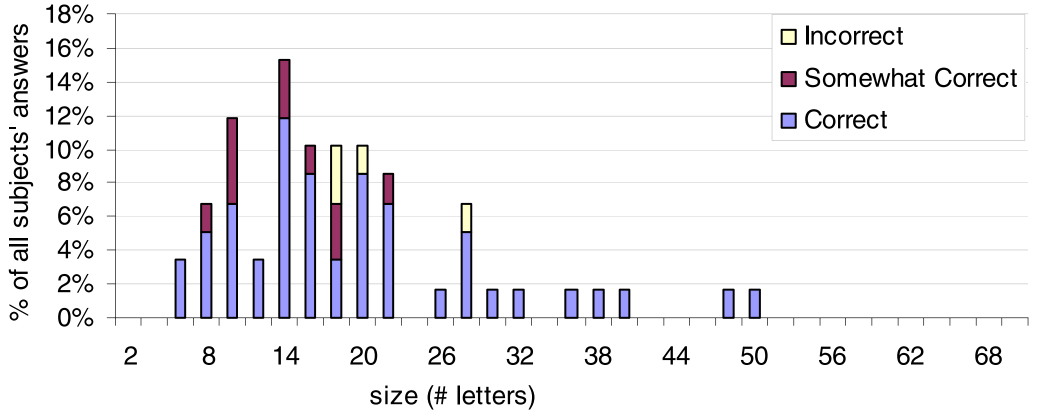


Figure 4-14 a. Distribution of title stop sizes using truncate-left method to identify web sites.

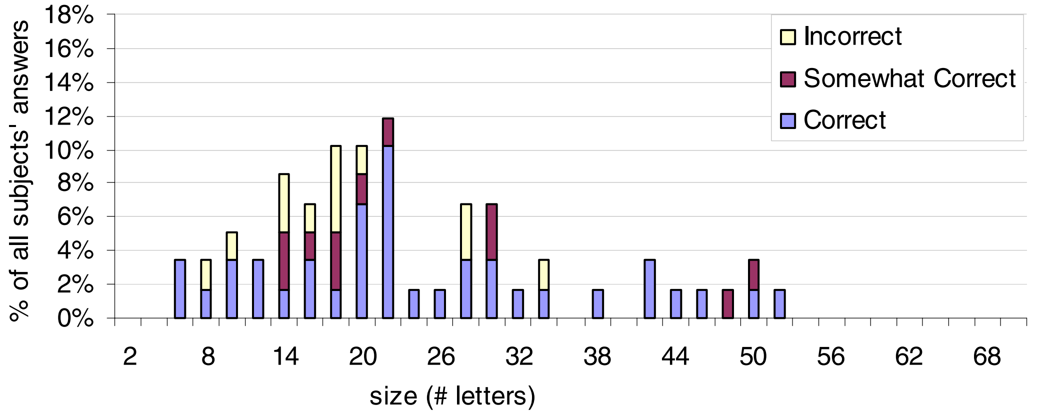


Figure 4-14 b. Distribution of title stop sizes using truncate-left method to identify exact pages.

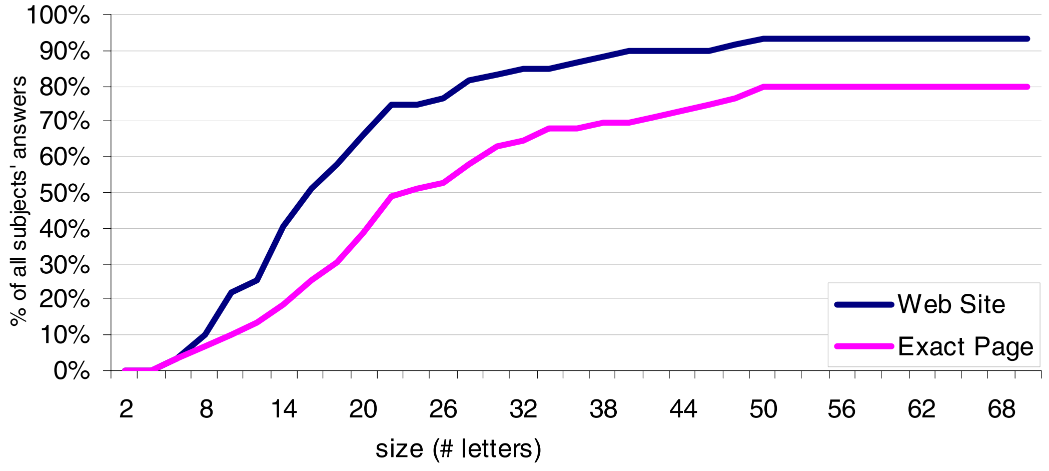


Figure 14 c. Running sums of title stop sizes using truncate-left method.

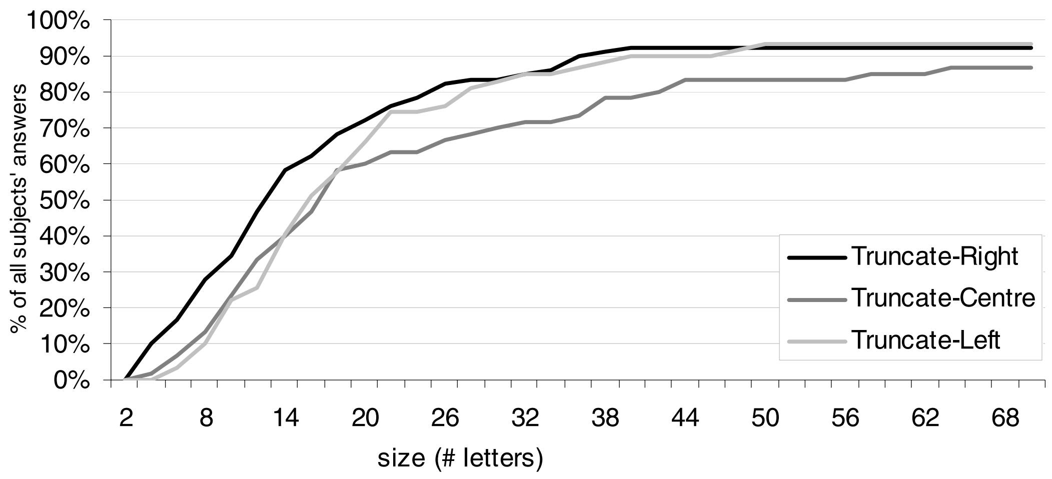


Figure 4-15 a. Running sums of title stop sizes for identifying web sites.

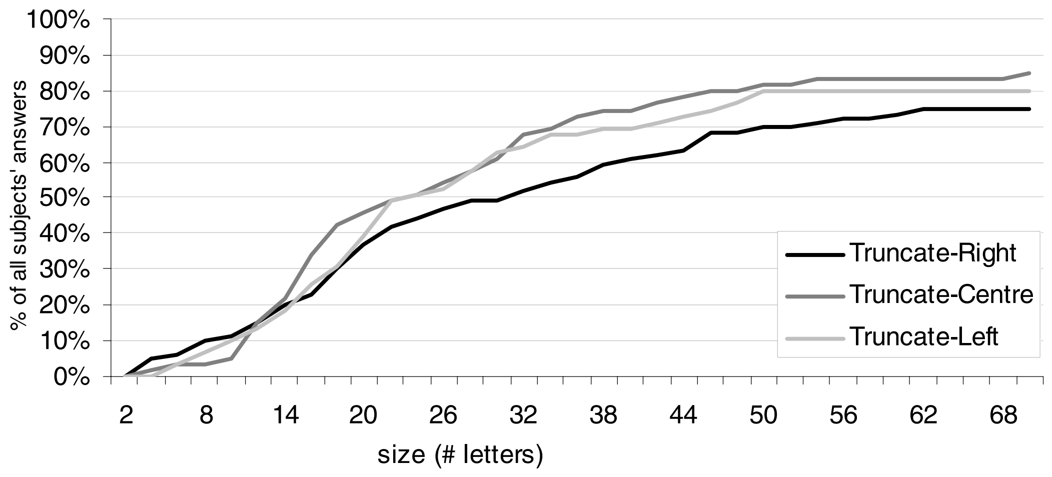


Figure 4-15 b. Running sums of title stop sizes for identifying exact pages.

Discussion. Of the three truncation methods, the commonly used truncate-right method stood out as decidedly different from the others. First, it was statistically significant as requiring the shortest size for identifying the web site. This is not surprising; many titles begin with the web site name, as in "University of Calgary - Department of Computer Science - Research" and the truncate-right method reveals that the beginning portion of the string.

For identifying the exact page, the discerning portion appears at the end of the title, as revealed by both the truncate left and center methods. Thus, truncate-right fairs poorly compared to the center and left methods (Figure 4-15b). Except at very low sizes, truncate-center is slightly favoured over truncate-left for identifying the exact page, as seen in the running sums of Figure 4-15b. This suggests that both prefix and suffix slightly re-enforce recognition. However, people need to see more letters of the title for identifying the exact page than the web site. For example, comparing the best-performing truncation methods between Figures 4-15a and 4-15b at 26 letters, we see that truncate-right gives us 82% recognition for web sites, while truncate-center gives only 54% recognition for exact pages; indeed we have to double the title length to 52 to bring recognition to 82% recognition rate.

As before, these distributions allow us to make recommendations for designing a revisitation list based on titles, as tabulated in Table 4-4. For example, for medium (60%) recognition, we need 15–20 letters (depending on the truncation method) for web sites, and 30–39 letters for exact pages. Unfortunately, no truncation method stands out as best for both web sites and exact page identification.

Table 4-4. Title recommendations for history list design.

Benchmarks	Identify Web Site			Identify Exact Page		
	Right	Centre	Left	Right	Centre	Left
Minimum (15%)	6	8	9	12	12	12
Low (30%)	8	12	12	18	16	18
Medium (60%)	15	20	18	39	30	28
High (80%)	25	42	28	>72	46	50

4.3.4 URL size

Research question 3 was: at what size threshold per truncation method do URLs have a reasonable chance of being recognized?

Before answering this question, we should mention that we included the standard 'http://' prefix in the URLs we presented to subjects as done in several existing history systems. Unfortunately, this meant that subjects shown the truncate-right and center URL did not see any useful portion of the beginning URL until after the 7 letters in 'http://'. In hindsight, we should have filtered off this prefix e.g., by showing 'www.ucalgary.ca…'

instead of 'http://www.ucal…'. (The same argument is not true for the 'www.' extension as it often differentiates intranet from internet pages). Consequently, we corrected our results. First, we subtracted 8 letters from the truncate-right size (because we increased the size in multiples of two, we could not subtract exactly 7 letters). Next, we subtracted 4 letters from the truncate-center size as this method reveals both the suffix and prefix: while not a great solution, it should be close enough for comparative purposes. We use and discuss only these corrected data in this thesis.

Results. As we did for titles, we will first look at the mean sizes at which the subjects were able to make identifications using the different methods. Figure 4-16 illustrates the mean sizes and standard deviations for identifying web site and exact page.

As before, we look to the stop size distribution for greater detail, where Figures 4-17, 4-18 and 4-19 a-c illustrate the distributions for each of the URL truncation methods. Figures 4-20 a-b collect Figures 4-17c, 4-18c, and 4-19c to help us visually compare the three truncation methods for identifying web site and exact page. As for thumbnails and titles, the cumulative distributions are based on the correct and somewhat correct responses only.

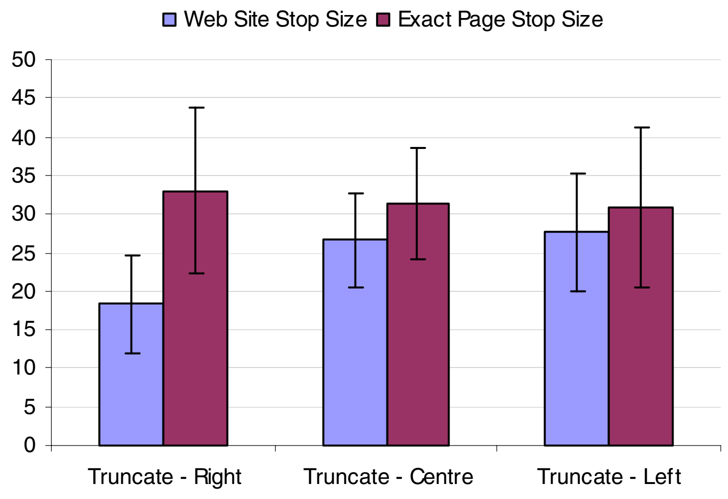


Figure 4-16. Subjects' mean stop sizes for identifying pages using URL truncation methods.

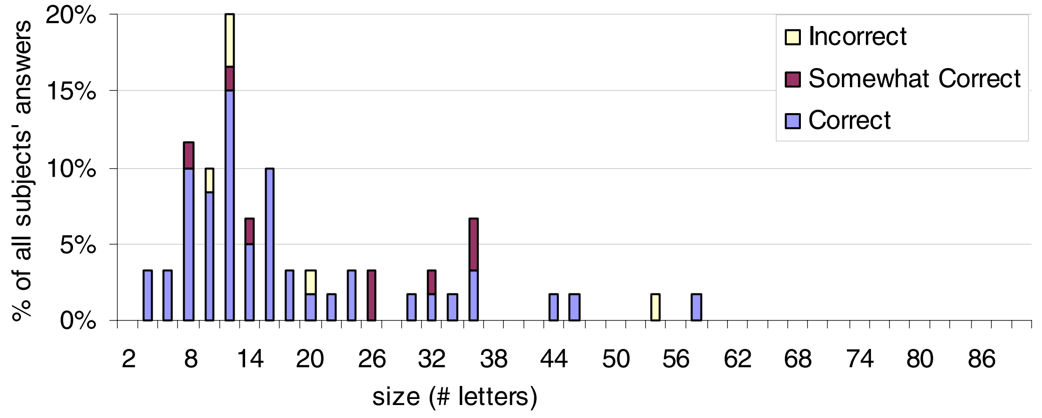


Figure 4-17 a. Distribution of URL stop sizes using truncate-right method to identify web sites.

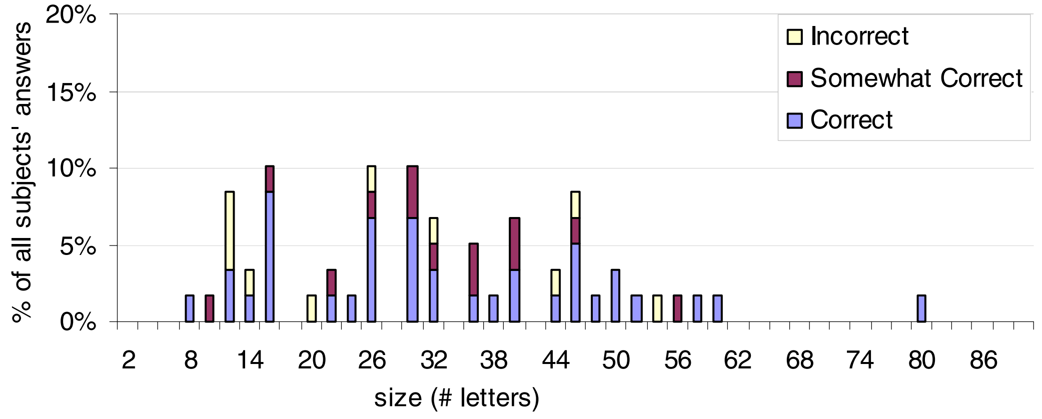


Figure 4-17 b. Distribution of URL stop sizes using truncate-right method to identify exact pages.

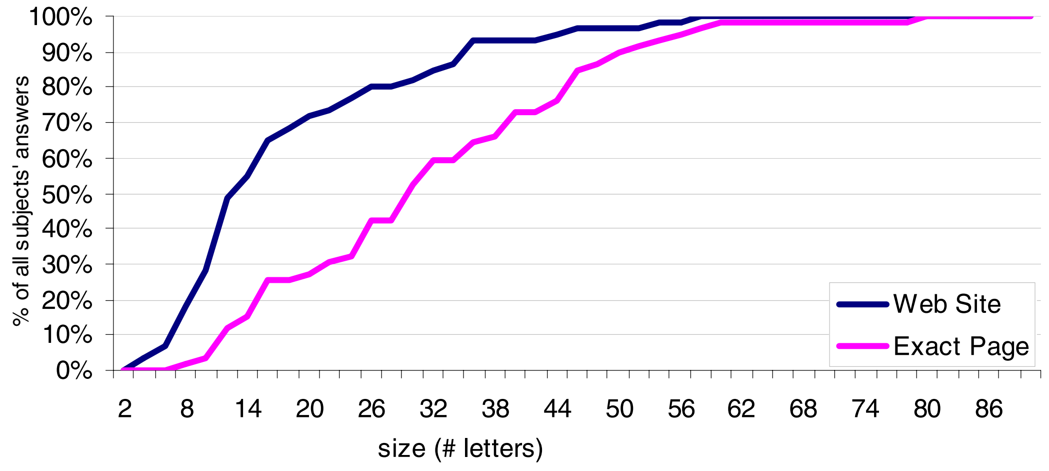


Figure 4-17 c. Running sums of URL stop sizes using truncate-right method.

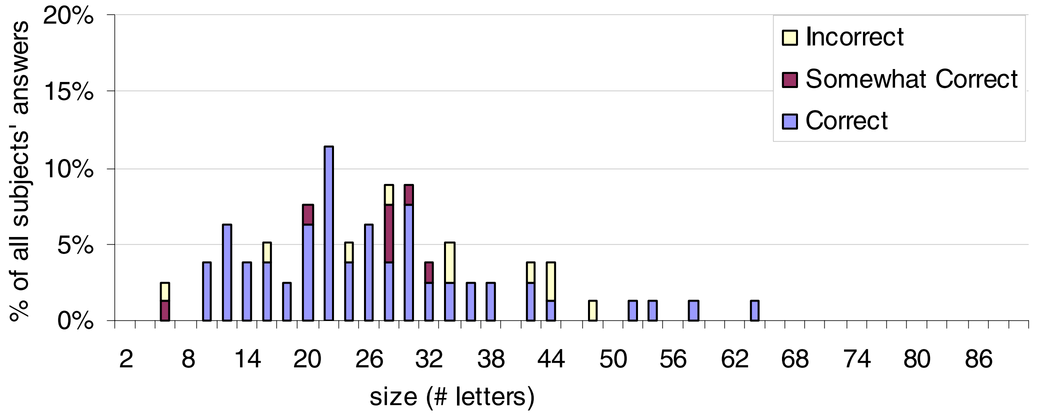


Figure 4-18 a. Distribution of URL stop sizes using truncate-centre method to identify web sites.

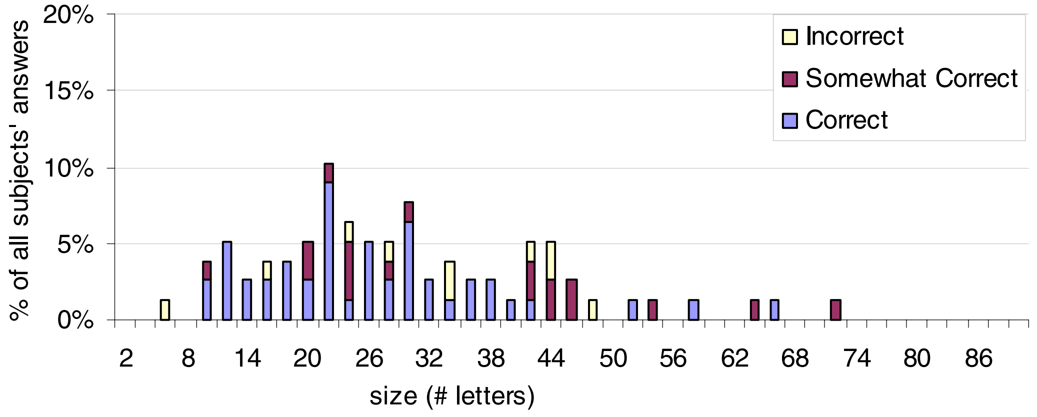


Figure 4-18 b. Distribution of URL stop sizes using truncate-centre method to identify exact pages.

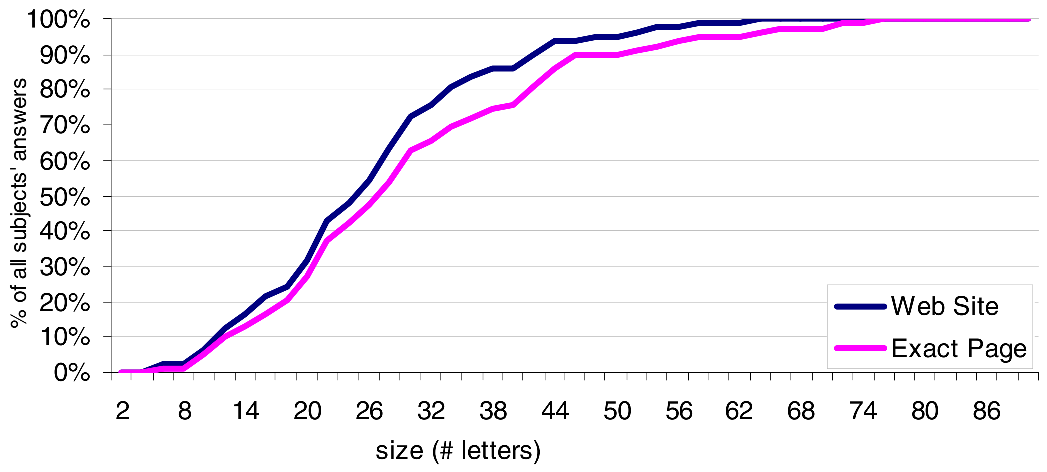


Figure 4-18 c. Running sums of URL stop sizes using truncate-centre method.

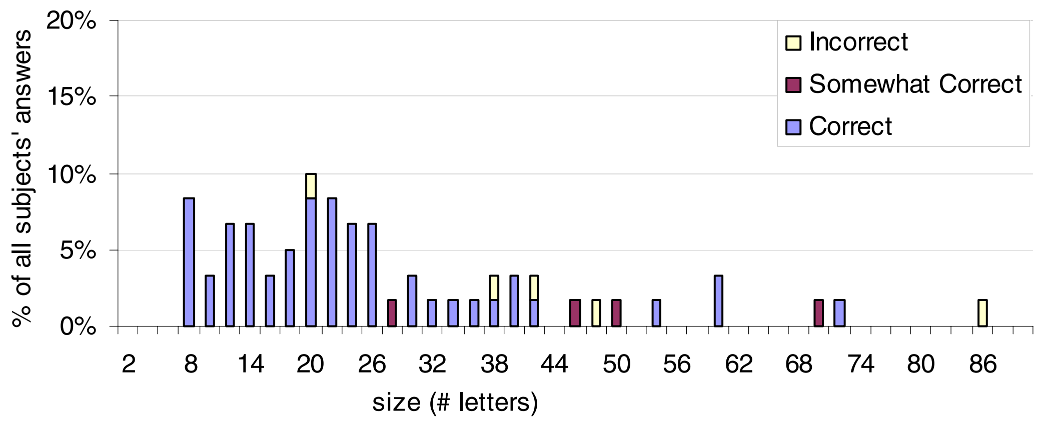


Figure 4-19 a. Distribution of URL stop sizes using truncate-left method to identify web sites.

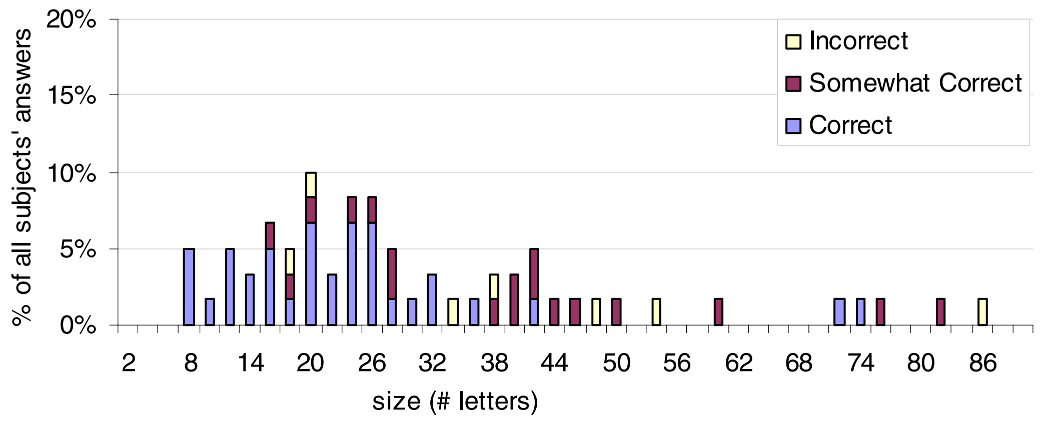


Figure 19 b. Distribution of URL stop sizes using truncate-left method to identify exact pages.

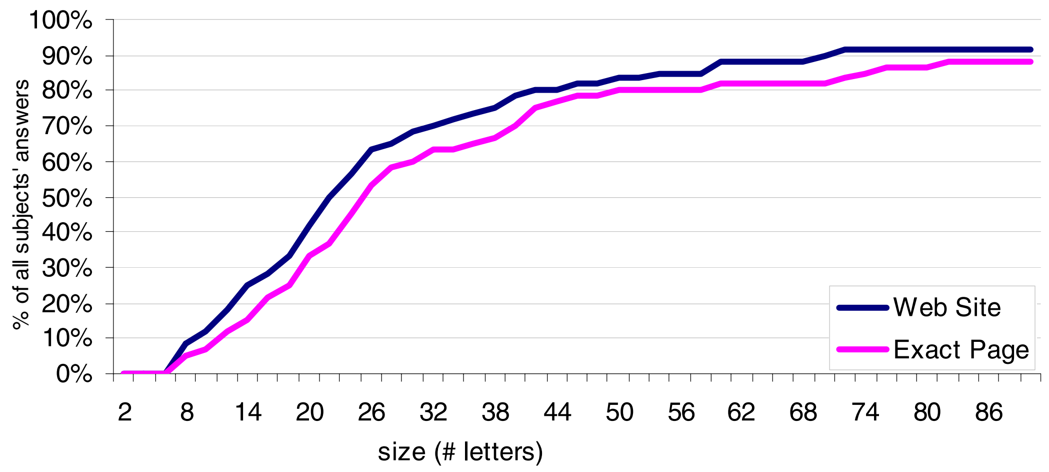


Figure 4-19 c. Running sums of URL stop sizes sing truncate-left method.

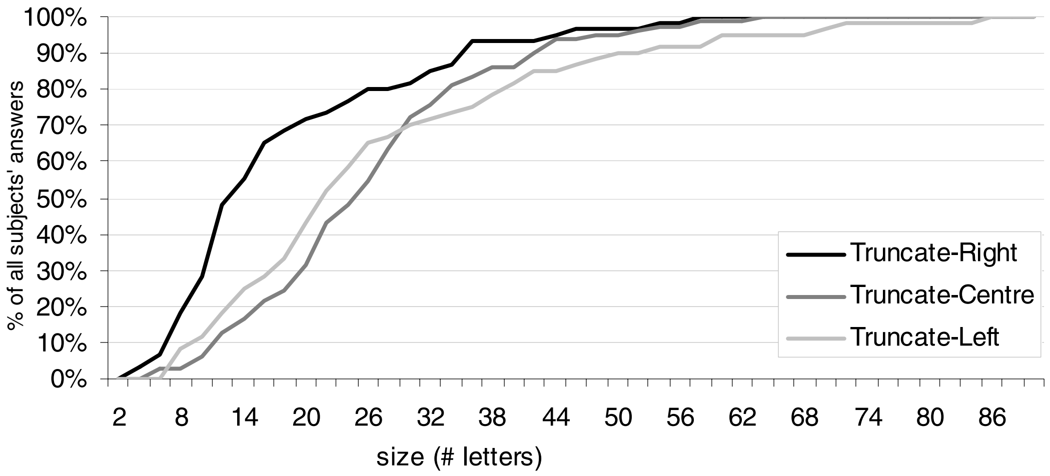


Figure 4-20 a. Running sums of URL stop sizes for identifying web sites.

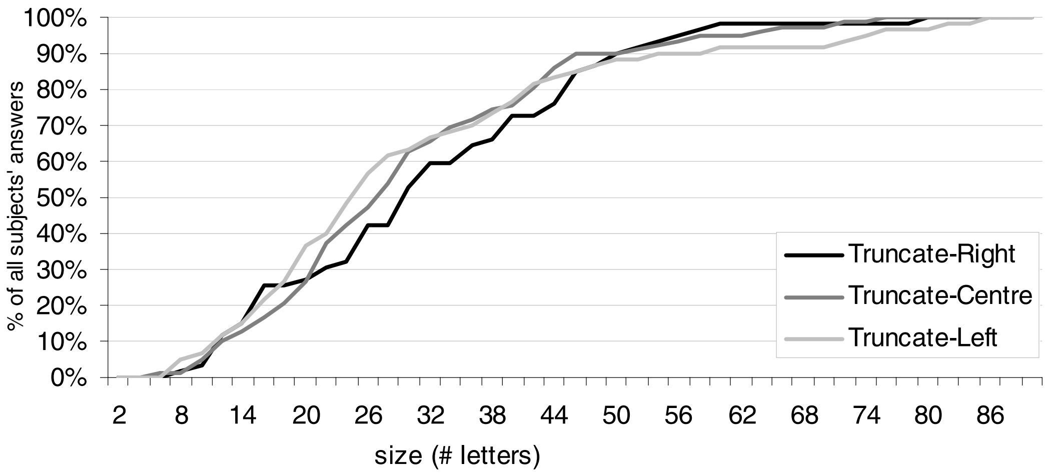


Figure 4-20 b. Running sums of URL stop sizes for identifying exact pages.

Discussion. For determining the web site, it is obvious that the beginning of the URL is an important cue. The name of the web site is often the same as the URL, such as www.microsoft.com, www.ucalgary.ca, etc. Indeed, the truncate-right stood out as the best method for web site identification.

For identifying exact page, the truncate-left method proved best. This too makes sense as the suffix is often a meaningful label for the exact page.

Overall, the distribution graphs reveal how much space must be dedicated to showing a URL in order to accommodate recognition. We will now use our benchmarks of 15%, 30%, 60% and 80% to make recommendations for minimum, low, medium and high recognition levels respectively. For minimum recognition, we need 8–14 letters to identify the web site and 14–16 letters to identify the exact page. For low (30%) recognition, we need 11–20 letters to identify the web site, and 19-25 letters to identify the exact page. For medium recognition, we need 16–29 letters for web sites and 30–43 for exact pages. Finally, for high recognition, we need 34–42 letters for web sites and 50–65 for exact page recognition. Table 4-5 summarizes these recommendations.

Table 4-5. URL recommendations for history list design.

Benchmarks	Identify Web Site			Identify Exact Page		
	Right	Centre	Left	Right	Centre	Left
Minimum (15%)	8	14	11	15	16	14
Low (30%)	11	20	17	25	22	19
Medium (60%)	16	29	25	43	34	30
High (80%)	34	43	42	58	65	50

4.3.5 Distribution of titles and URLs on the Web

Research question 4 asked: what is the distribution of length for titles and URLs from pages typically found on the Web?

Results. The collected data provided two different samples of pages to analyze. The first analysis included the pages selected for the subjects' trials, i.e. 30 pages for each of the 20 subjects, totalling 600 pages. From this sample, the mean title-length was 31 letters, with a standard deviation of 20.

The second sample used all of the pages from the history records submitted by the subjects. This included about 9200 pages. The mean title-length from this sample was also 31 letters, with a standard deviation of 22. The distribution graphs for the two samples were also nearly identical. Figures 4-21a-b shows the regular and cumulative distributions for the 9200 pages sample of titles. Note that the pages without titles (accounting for 30% of the total) are included as length 0.

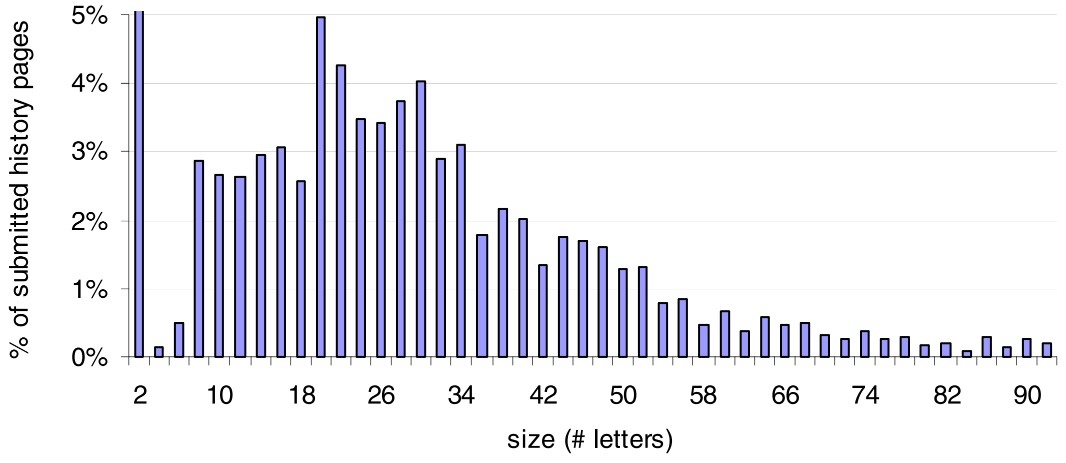


Figure 4-21 a. Distribution of title lengths taken from all of the subjects' submitted histories.

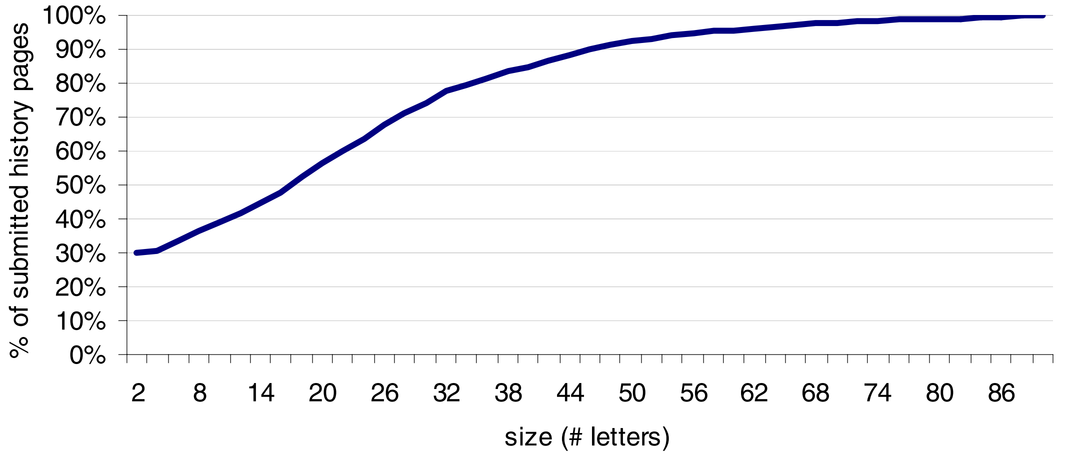


Figure 4-21 b. Running sum of title lengths taken from all of the subjects submitted histories.

Discussion. First, it was surprising that 30% of these pages did not have titles. This is much higher than Cockburn's finding (2000) that 5% of pages are missing titles. While these studies had a similar number of subjects (Cockburn's 17 to our 20), Cockburn had a much larger total number of pages—17242 distinct URLs compared to the 9200 URLs in this study. We suspect this difference is due to several factors. While Cockburn's counted one page for a page composed of frames, our system counted an entry for each frame. Thus, these additionally pages counted as pages without titles. This included popup advertisements that are common on commercial web sites. Also, many of the subjects' pages were taken from web-based mail and discussion forums and often had no titles. We are apprehensive about making strong claims about this finding. We expect that the true number of titles pages is likely to be somewhat grater than 5%, but much less than 30%. One possible route for further investigation is to enlist the help of a major search engine such as Google. Their database contains information on over 1.3 billion web pages. It seems like an easy task for them to find out how many of these pages are untitled, and to see how it compares to our 30% and Cockburn's 5% findings.

This sample of titles provides important information about the prevalence of truncation. Specifically, we have an idea of how much of a title is hidden at various truncation sizes. For example, if a system displays only the first 20 letters, then only 55% of the titles will fit completely. Thus, users will have to make decisions based on incomplete information for almost half of the items.

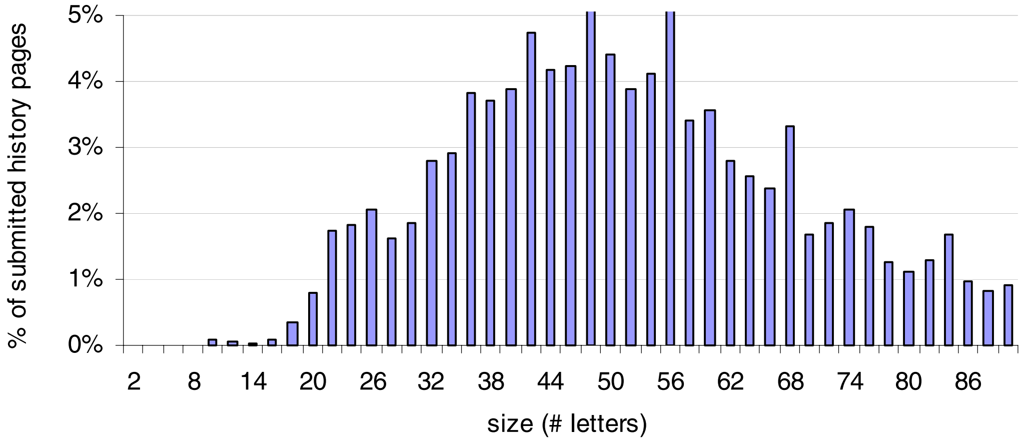


Figure 4-22 a. Distribution of URL lengths taken from all of the subjects' submitted histories.

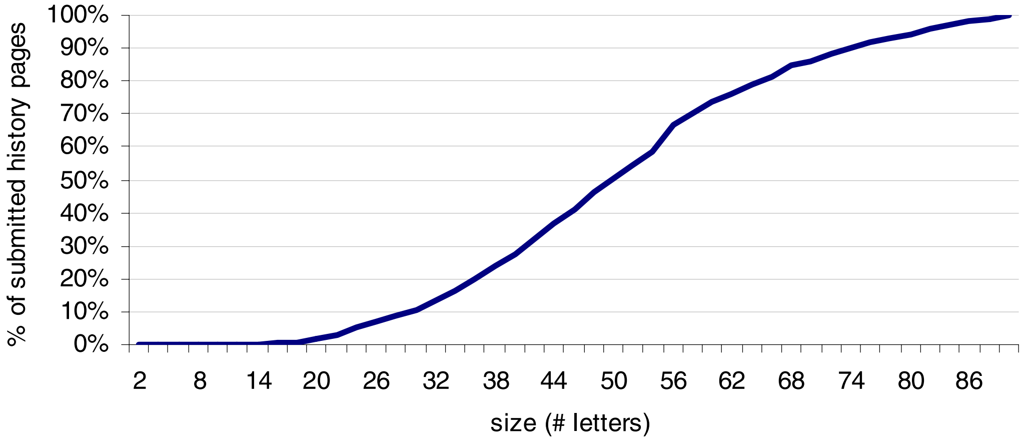


Figure 4-22 b. Running sums of URL lengths taken from all of the subjects' submitted histories.

For URLs, we will start by looking at the statistics for all of the URLs used in the study. The average URL-length was 47 letters, with a standard deviation of 22. Figure 4-22a plots the distribution, and Figure 4-22b plots the distribution as a running sum.

Discussion. Again, the distribution allows us to see how often URLs will appear truncated to the user. For example, if the system displays only the first 20 letters of a URL, only 1% of the URLs will fit completely. Thus, the user will have to make decisions with information missing from virtually all of the entries.

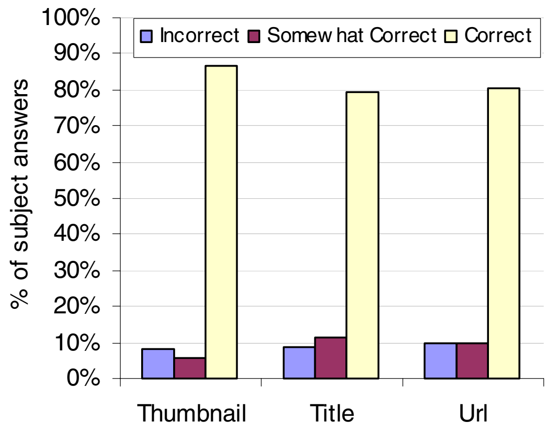
4.3.6 Representation verification

This section compares how correctly the subjects were able to identify the pages with thumbnails, titles, and URLs. In this analysis, the data for the three different title and URL truncation methods have been aggregated into general 'Titles' and URLs' categories. We based the scoring on how well the subjects were able to correctly identify both the page type and the exact page when shown the representation. There were three options for scoring: 0- Incorrect, 1- Somewhat Correct, and 2- Correct.

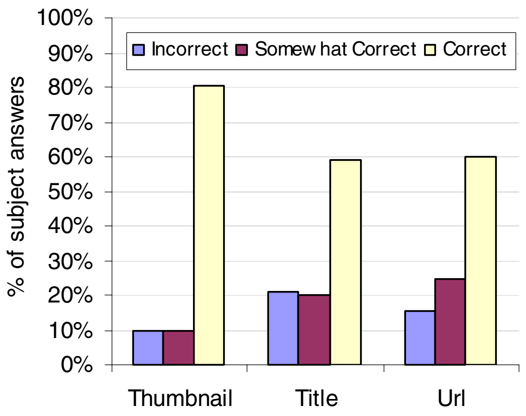
Results. Figure 4-23a shows how the subjects fared for identifying the web site. Statistically, there was no difference between the different error rates of the various representations (ANOVA, $p \geq 0.05$, F=1.21).

Figure 4-23b shows the results for identifying the exact page. Subjects performed better with thumbnails than with titles or URLs (ANOVA $p \leq 0.05$, F=12.21)(TTEST $p \leq 0.05$).

Discussion. Figure 4-23a illustrates that the subjects were quite accurate at identifying the web site with fewer than 10% of their answers being incorrect. The subjects' identifications were completely correct for ~80% of the pages, for all representations (thumbnail, title, URL).



a.



b.

Figure 4-23. Subjects' accuracy at identifying (a) web sites and (b) exact pages.

The exact page identifications were not as accurate when using titles and URLs – 60% compared to 80% for the web site. The subjects using thumbnails however, achieved 80% for both identifying web site and exact page. Thus, the thumbnails were effective for identifying both the web site and the exact page. This supports the hypothesis that thumbnails are a useful representation for a history list. It also suggests that titles and URLs, the standard history list representations, are not as effective as one would like. If users only have a 60% chance of recognizing the exact content from a title or URL in the history list, they may not be motivated to invest the extra work it takes to operate the history list (opening, scrolling, closing) in order to track down a page. Simply put, it is not worth the effort to switch to a history list containing items of which only 60% are recognizable.

4.3.7 Subjects' representation ratings

Along with performance measures, each subject rated how well the representation 'captured' the content of the page. The ratings ranged from 0- Does not represent page content at all to 4- Instantly recognizable.

Results. Figure 4-11 shows the results, converted to percentages.

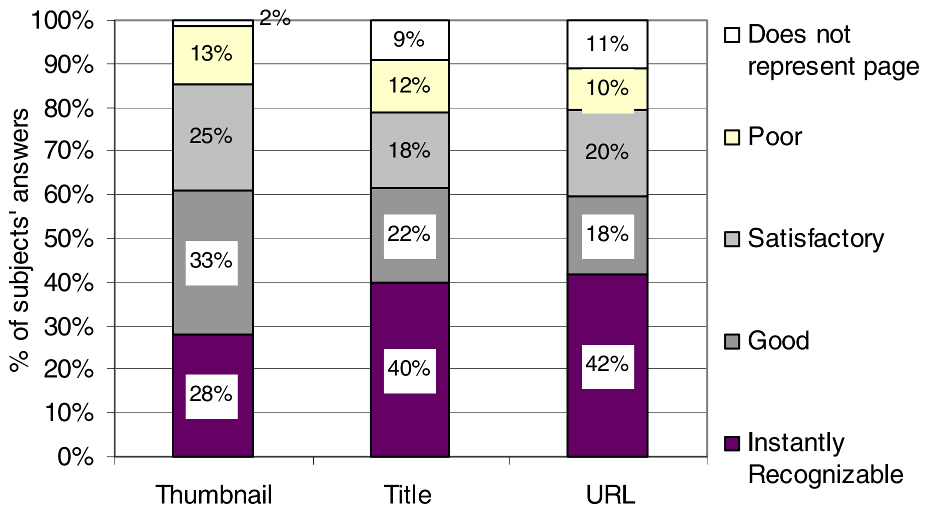


Figure 4-24. Subjects' ratings of the thumbnails, titles and URLs.

The subjects rated 15% of all thumbnails and 21% of titles and URLs as being poor or worse. Thus, thumbnails received the fewest less than satisfactory ratings.

There are some minor differences between thumbnails vs. titles/URLs. While all had ratings of ~60% in the good or higher category, there were fewer thumbnails in the instantly recognizable category compared to titles/URLs. However, very few thumbnails (2%) were in the 'does not represent the page' category when compared to titles/URLs (~10%).

Discussion. These results suggest that thumbnails are a somewhat better representation than titles or URLs. Titles and URLs were somewhat bipolar, either instantly recognizable or quite poor, where as thumbnails had less extreme ratings.

As seen in the graph, 85% of all thumbnails and 80% of both titles and URLs were considered satisfactory representations or better. Again, the issue of missing titles must be accounted for. While this experiment found that 30% of pages did not have titles, it is more likely that in practice this occurs for only about 5% of pages (as was found in the Cockburn and McKenzie, 2000 study). Regardless of the percentage, if the title is missing, then some other representation must be used, which may or may not be meaningful to the user.

4.4 Conclusion

This experiment has touched upon many issues for the design of the history list. First, the subjects' thresholds for recognizing pages by thumbnail, title, and URL allowed us to make the following recommendations:

Table 4-6. Size recommendations for each representation.

	Size required for identification														
Recognition rate	**Thumbnails**		**Titles**						**URLs**						
	web site	***exact page***	***web site***			***exact page***			***web site***			***exact page***			
			right	centre	left	right	centre	left	right	centre	left	right	centre	left
Minimal: 15%	32^2	48^2	6	8	9	12	12	12	8	14	11	15	16	14
Low: 30%	48^2	80^2	8	12	12	18	16	18	11	20	17	25	22	19
Medium: 60%	96^2	144^2	15	20	18	39	30	28	16	29	25	43	34	30
High: 80%	160^2	208^2	25	42	28	–	46	50	34	43	42	58	65	50

When subjects utilized thumbnails, they identified web sites mainly by relying on colour and layout, and identified exact page content by reading the text that became legible at large sizes.

The accuracy of the subjects' identifications revealed that thumbnails were most effective for recognizing exact page content.

The subjects' ratings of the representations suggested that thumbnails were often satisfactory or good, while titles and URLs were often instantly recognizable or poor.

All of these findings have direct application to the design of a history list. However, these results are also important for web page designers. It is in their best interest to design pages that can be effectively recognized and therefore revisited easily. Web pages should have specified, short but well named titles. The URL should be descriptive, yet not overly long. For thumbnail images to be effective, pages throughout a single web site should have a consistent layout and colour scheme. Large text for important headings will be useful for thumbnail images as well.

Finally, these findings are based on a technically oriented subject group (all were students of Computer Science). Thus, it is foreseeable that a more general population will not be as responsive to URLs or titles that contain technical rather than content related terms (e.g., 'Index.html' is a home page). However, there is no clear reason why the thumbnail findings could not apply to a general population.

5. Technical Issues

While the Unified History system described in Chapter 3 seems relatively straightforward, there were a surprising number of technical issues that had to be overcome. Some concerned strategies for implementing novel features; others arose from having to overcome limitations of how one can tap into Internet Explorer. Still others included problems with capturing effective web histories. In this chapter I outline some of these problems and the workarounds that I used. I also indicate useful ways that Internet Explorer (and the web itself) could be altered to make advanced revisitation facilities effective. These problem identifications and solutions should be useful for other researchers and practitioners who wish to replicate our system or create their own alternative revisitation system.

The major technical issues that arose when designing and implementing the system, listed below.

- window vs. global history
- communicating between facilities
- frames
- redirection pages
- thumbnail implementation;
- tracking history
- implementing the explorer bar

5.1 Window vs. global history

5.1.1 The problem

All popular browsers use a window-based Back system; i.e., Back only covers the navigation events that occurred in the current window within the current browsing session. When users have multiple windows open, this makes it difficult to return to pages using Back, as it only takes the user to pages viewed in that particular window.

Having multiple windows open can occur without the user's consent, as some web sites have links that open up new browser windows. Thus, the user suddenly has multiple windows open. As described in Chapter 2, the standard browsers use a stack for each window. When a page is loaded in the new window, it is the only page on the new stack. Thus, the Back button is 'greyed' out, even though the user has visited numerous pages before the current page. To find these previous pages, the user must switch to the other window and click Back. Web Usability writer Jakob Neilson listed this issue as #2 in “The Top Ten New Mistakes of Web Design” in his May 30, 1999 Alertbox (www.usableweb.com) column.

Our suggestion throughout this thesis is to do away with the stack model for Back/Forward in favour of modeling them on recency. This model ensures that the user can reach any previously visited page simply by repeatedly pushing Back. This is true even if multiple windows are open, as each window's Back and Forward now work the same.

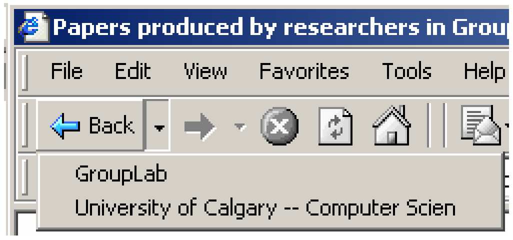
In order to implement this model, rather than have a different history list for each window, we need an all-inclusive global history. We will now go through an example that illustrates the difference between having Back/Forward work on a stack and having them use a global history. In this example, a user named ‘John’ navigates to the following pages, in order:

- University of Calgary Homepage
- The Grouplab Research Home Page
- Papers By Grouplab

➢ (New Window) A paper abstract for “Integrating Back, History and Bookmarks”

John visited the first three pages in a single web browser window, and then selected the ‘Abstract: Integrating Back…’ paper, using the ‘Open in new window’ option provided by the browser. John now has two windows open. If John presses the Back button drop down list in Window 1 (Figure 5-1a), he can select the Grouplab or the University of Calgary page. However, he cannot press Back in Window 2 (Figure 5-1b), as the Back button is ‘greyed out’. Also John cannot press Forward in Window 1 (Figure 5-1a) to visit the Abstract page, since that page was visited in Window 2.

a. Window 1


Papers produced by researchers in Grou
File Edit View Favorites Tools Help
Back
GroupLab
University of Calgary -- Computer Scien

b. Window 2

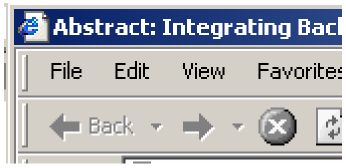
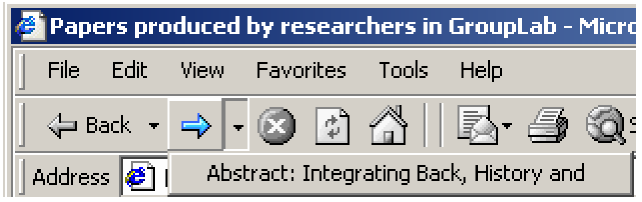


Figure 5-1. Pages visited in Window 2 are not accessible from Window 2 via back/forward. Also, the page in Window 2 is not accessible in Window 1 via forward.

a. Window 1


Papers produced by researchers in GroupLab - Micro
File Edit View Favorites Tools Help
Back
Address
Abstract: Integrating Back, History and

b. Window 2

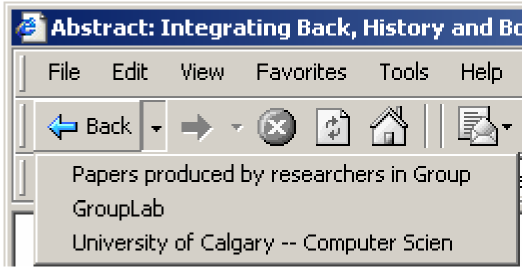

Abstract: Integrating Back, History and Bo
File Edit View Favorites Tools Help
Back
Papers produced by researchers in Group
GroupLab
University of Calgary -- Computer Scien

Figure 5-2. All pages are accessible from each window’s back/forward buttons.

In Figure 5-2a-b, we see what would happen if the browser used a global rather than window-based history. In Figure 5-2a, John can press Forward to visit the Abstract page. In Figure 5-2b, Window 2’s back button can access the previous pages.

As mentioned in Section 3.2.1, there are trade-offs to using either global history or window history. The advantage of global history is that one does not need to remember which page was visited in a particular window. The disadvantage is that it is difficult to carry out multiple browsing tasks without the history of each task intertwining.

5.1.2 Implementing global history

Implementing global history requires all of the browser windows to be aware of all navigations. This was a difficult challenge for Unified History, as the Internet Explorer generates events only to the individual windows. Even more difficult, there is no trivial approach to having a particular browser window know that there are other browser windows open.

The solution used for Unified History was to create a special history server, so each browser becomes a client connected to it through a special communication layer. When the user navigates, the browser window client sends this information to the server process, which then relays it back to the other browser window clients. Figure 5-3 illustrates this idea.

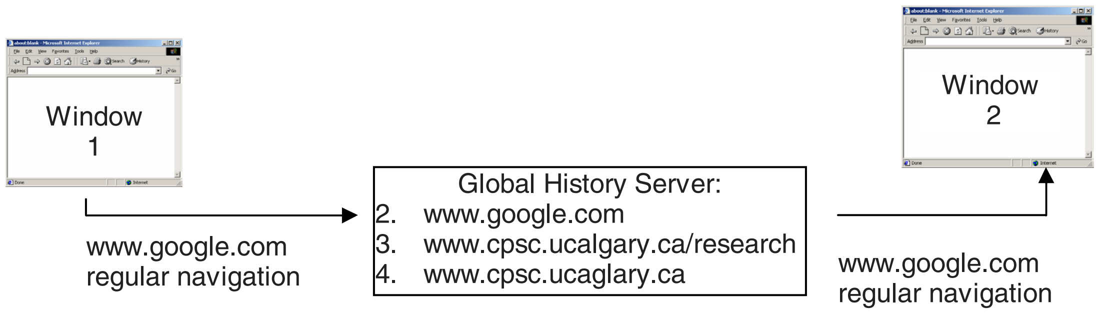

Window 1
www.google.com regular navigation
Global History Server:
2. www.google.com
3. www.cpsc.ucalgary.ca/research
4. www.cpsc.ucaglary.ca
Window 2
www.google.com regular navigation

Figure 5-3. When the user navigates in Window 1, the window communicates the event to a server process that passes it on to the other windows.

5.1.3 Suggestion to browser designers

The task of implementing global history was difficult due to the lack of communication between Internet Explorer windows. Specifically, the browser windows execute as separate processes, unaware of each other's events. Thus, I needed to implement a server process to coordinate messages between the windows.

A useful extension to web browsers would be an intrinsic method of querying and communicating between the various open browser windows. This would eliminate having an extra server process to facilitate this message passing.

5.2 Communication between facilities

5.2.1 The problem

Since the various revisitation facilities are completely separate entities, there is no support for making them communicate with each other. Having the various browser windows communicate with each other is just one example of the communication required to integrate these revisitation facilities. The history list also needs to be aware of the navigations occurring in all of the browser windows. Otherwise, the history list would not include pages navigated in other windows. Also, the history list needs to display thumbnail icons that have been captured in other browser windows. Thus, either directly or through the server process, all of the revisitation facilities need to communicate with each other.

5.2.2 Implementing communication between facilities

As described in the previous section, the ***Unified History Server*** is responsible for coordinating communication between the various browser windows. There is also a global ***Thumbnails File***, which stores the thumbnail images that have been captured. The other components involved in the system are created with each new browser window. These include the ***Back/Forward buttons*** and ***History Explorer Bar*** which are visible to the user in the browser window. Unseen to the user is the ***Browser Helper Object***, which has the direct communication with the browser window's events. Consequently, the ***Browser Helper Object*** informs the ***Unified History*** server about the navigate events and handles the thumbnail capturing.

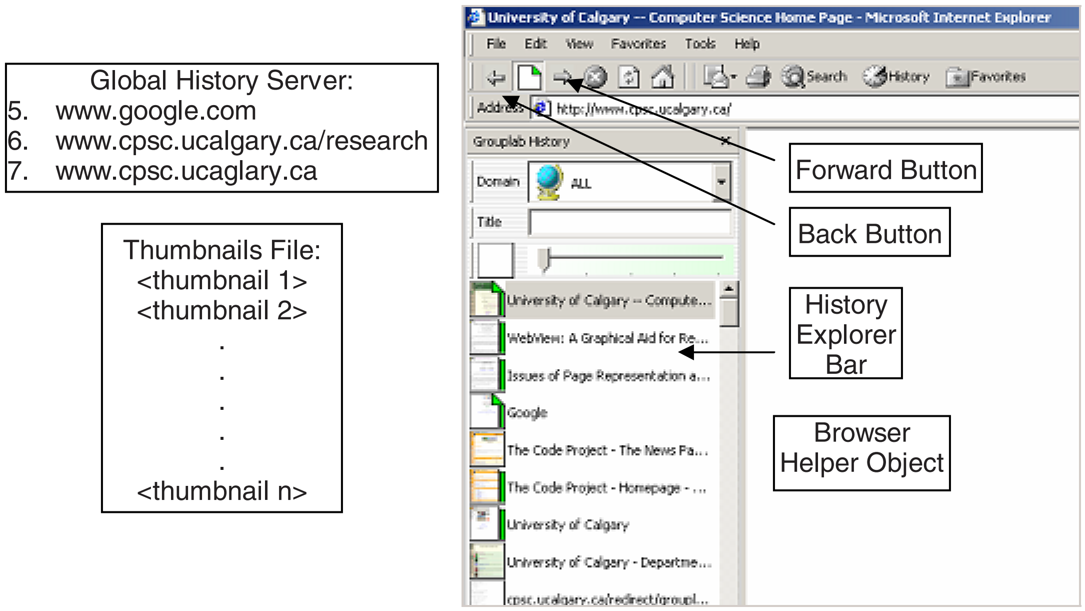

Global History Server:
5. www.google.com
6. www.cpsc.ucalgary.ca/research
7. www.cpsc.ucaglary.ca
Thumbnails File:
<thumbnail 1>
<thumbnail 2>
<thumbnail n>
University of Calgary -- Computer Science Home Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Search History Favorites
Address http://www.cpsc.ucalgary.ca/
Grouplab History
Domain ALL
Title
University of Calgary -- Compute...
WebView: A Graphical Aid for Re...
Issues of Page Representation a...
Google
The Code Project - The News Pa...
The Code Project - Homepage - ...
University of Calgary
University of Calgary - Departme...
cpsc.ucalgary.ca/redirect/groupl...
Forward Button
Back Button
History Explorer Bar
Browser Helper Object

Figure 5-4. Illustration of the various Unified History components.

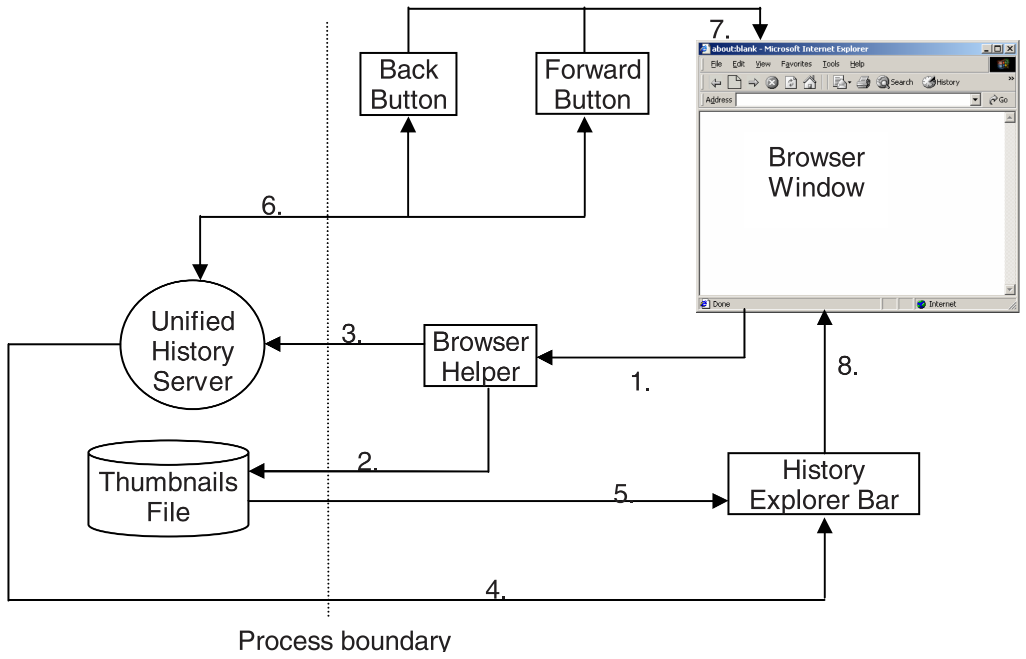


Figure 5-5. Communication between the various Unified History components.

Figure 5-5 illustrates an example of how these components interact with each other. Let us say that John opens the first instance of Internet Explorer. This automatically creates a ***Browser Helper*** object, and creates the ***Back/Forward*** buttons that appear on the toolbar. Since this is the very first instance of Internet Explorer, a new ***Unified***

History Server object is also created automatically. If John had opened a second browser window, the Unified History server would already be running. By default, the ***History List*** is not open. When John clicks on its icon to open the history list, the ***History Explorer Bar*** component is created.

As John navigates pages, the ***Browser Helper*** object receives the various navigation events from the browser window (Figure 5-5, Link 1). Responding to these events, the browser helper object captures the thumbnail images of the various pages and saves these to the ***Thumbnails file*** (Link 2). Section 5.4 describes how the thumbnails are captured and saved. When the thumbnail file is successfully saved, the helper object notifies this event to the ***Unified History Server*** (Link 3). The server than tells the ***History Explorer Bar*** (Link 4), so the explorer bar can load the thumbnail (Link 5) and display it to the user.

The ***Browser Helper*** also passes on the navigation events to ***Unified History Server*** so that the server can keep track of the page ordering for the back/forward buttons. The server then communicates to the ***Back/Forward*** buttons (Link 6) in order to 'grey them out' when there are no logical Back or Forward pages.

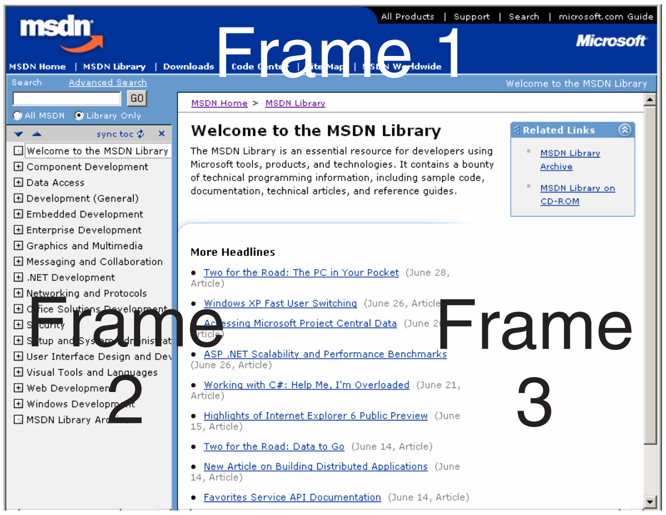
When John clicks on either ***Back*** or ***Forward***, these buttons ask the ***Unified History Server*** for the URL (Link 6) that is either backward or forward in the global history. The buttons then tell the browser window to navigate to the given URL (Link 7). Since John has the ***History Explorer Bar*** open, the server also sends the event to the ***History Explorer Bar*** (Link 4) so that the listing can highlight the page as a move back or forward.

The only components that directly control the web browser are the ***Back/Forward*** buttons (Link 7) and the ***History Explorer Bar*** (Link 8). When John presses Back or Forward, or selects a page in the history list, these components instruct the browser window to navigate to the given page. This is treated differently than a regular navigation event (clicking a link or typing an address), so that 1) the ***Unified History Server*** can maintain the proper Recency ordering and 2) the **History Explorer Bar** can highlight the page where it occurs in the list instead of adding the page name to the top of the list.

5.2 Frames

5.2.3 The problem

Frames, which first appeared in 1996, are a controversial topic on the World Wide Web. Frames are web pages that consist of multiple pages. In Figure 5-6, we see a web page (http://www.msdn.microsoft.com/library) made up of three frames. This page utilizes frames so that the user can browse different articles (Frame 3) without losing his/her place in the table of contents (Frame 1).


msdn
All Products | Support | Search | microsoft.com Guide
Microsoft
MSDN Home | MSDN Library | Downloads | Code Center | Site Map | MSDN Worldwide
Frame 1
Search
Advanced Search
GO
All MSDN
Library Only
sync toc
Welcome to the MSDN Library
Frame 2
Frame 3

Welcome to the MSDN Library
Component Development
Data Access
Development (General)
Embedded Development
Enterprise Development
Graphics and Multimedia
Messaging and Collaboration
.NET Development
Networking and Protocols
Office Solutions Development
Security
Setup and System Administrat
User Interface Design and Dev
Visual Tools and Languages
Web Development
Windows Development
MSDN Library Archive

MSDN Home > MSDN Library

Welcome to the MSDN Library

The MSDN Library is an essential resource for developers using Microsoft tools, products, and technologies. It contains a bounty of technical programming information, including sample code, documentation, technical articles, and reference guides.

Related Links

- MSDN Library Archive
- MSDN Library on CD-ROM

More Headlines

- Two for the Road: The PC in Your Pocket (June 28, Article)
- Windows XP Fast User Switching (June 26, Article)
- Accessing Microsoft Project Central Data (June 2[illegible] Article)
- ASP .NET Scalability and Performance Benchmarks (June 26, Article)
- Working with C#: Help Me, I'm Overloaded (June 21, Article)
- Highlights of Internet Explorer 6 Public Preview (June 15, Article)
- Two for the Road: Data to Go (June 14, Article)
- New Article on Building Distributed Applications (June 14, Article)
- Favorites Service API Documentation (June 14, Article)

Figure 5-6. Microsoft's MSDN Library page – utilizing frames (with frames labelled).

When first introduced, frames pages led to many usability problems. Jakob Neilson named using frames as the #1 Mistake in Web Design in his May 1996 Alertbox article. First, Netscape 2, which was still popular at the time, had problems with the way Back and Forward managed frames pages. In particular, this led to the following situation. The user would navigate to a page containing frames, visiting various different pages contained on the site. Then the user would click back, expecting to move to a previously seen page of the site. Instead, the browser would lead back to the page visited before the frames site. The browser did not record any of the navigations in the frames pages.

The second problem with frames is that frames pages are difficult to bookmark. Often, the user wants to make a bookmark to return to a specific state of all three frames.

Unfortunately, clicking 'Add Bookmark' makes a bookmark for the current URL, which refers to the initial state of the frames rather than the current one.

Unfortunately, these two problems had to be dealt with for implementing Unified History. Even though current browsers' Back/Forward buttons now work correctly for frames pages, its mechanism was not available to end programmers, which I learned through discussion with Microsoft’s Internet Explorer developers. Thus, in order to re-implement Back/Forward, I had to overcome the same problems as the browser designers back in 1995.

5.2.4 Implementing proper frames behaviour

To solve the frames problem, I used the following algorithm:

1. Treat the initially loaded frames as all being a single web page – that is ignore the events loaded for each of them and only have a history entry for the complete page. Thus, when the user first visits the frames site, the history list will contain one entry for the page, rather than separate entries for each of the individual frames.
2. When navigations occur inside the frames site, make new entries for each of these navigations. This way, the history list contains an entry for each of the user’s subsequent navigation actions.

This algorithm produced the ‘correct’ behaviour in that the history list accurately reflects the user’s navigations. However, it is ‘incorrect’ in that when the user clicks Back, only the content frames appear, but not the surrounding frames. It is not clear how Internet Explorer is able to solve this problem – it likely involves recording what frames surrounded each particular frame when the user last saw it. I settled a compromise solution for three reasons. First, the user is likely interested in the content rather than the surrounding menu. Second, when the user clicks Back enough times to return to the first entry of the web site, the frames appear in their initial state. Finally, while necessary in a commercial system, I felt this level of detail was not relevant for research prototype.

5.2.5 Suggestion to Browser Designers

This task of dealing with frames was in many ways 'reinventing the wheel'. This could have been avoided if the browser included better support for frames sets—groups of frame pages. Currently the event and history model deals exclusively with individual pages. Thus, it is very difficult to respond to navigations where many frames pages are changing. However, frames support is something the browser designers seem hesitant to improve. Due to the hassles frames cause for both developers and end users, more and more web sites are moving away from using frames. Thus, the browser developers have little incentive to invest effort in solving the frames issues.

5.3 Redirection Pages

5.3.1 The problem

Various web pages immediately redirect the browser to navigate to a different page, usually (but not always) displayed in the same window. This occurs as ways for sites to redirect people to new addresses from a no longer used one, or to bring up a site specific error page (e.g., if the page does not exist), or to show advertisements in another window. For example, if a person navigates to page 1a, that page will automatically navigate to another page 1b. Thus, two pages are visited even though a single link was selected.

Even the modern browsers' standard Back/Forward buttons have trouble with redirections. When the user traces through history with Back, accessing a redirect page takes the user right back to the just seen page. Thus, the user is stuck, for he or she cannot get past the redirect page. He or she must use the Back button dropdown menu (Figure 5-7) to bypass the redirect page.

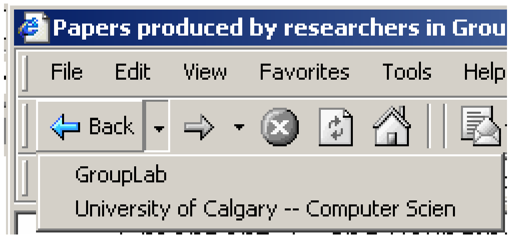

Papers produced by researchers in Grou
File
Edit
View
Favorites
Tools
Help
Back
GroupLab
University of Calgary -- Computer Scien

Figure 5-7. Drop-down list on back button allows user to skip over redirect pages.

Redirection pages are problematic for the history list in that they occupy space in the list, yet have no useful content. In most cases, the user would simply prefer to see the redirector's target page in the list.

We were unable to overcome this problem, for there is no easy way to differentiate redirection navigations from regular page navigations. Thus it is impossible to filter out only the redirection pages.

5.3.2 Suggestion to browser designers

Redirection pages should be excluded from the navigation events or differentiated from regular navigations. Also, the Back/Forward buttons should not be 'blocked' by redirect pages. This will allow users to click Back to return to the pages viewed before the redirection, without having to bypass over the redirection page.

5.4 Thumbnail implementation

Chapter 4 found that thumbnails are useful cues for recalling web pages. While titles and URLs are supplied for free with existing browsers, there are many technological issues with obtaining web page thumbnails.

5.4.1 Aspect ratio and clipping

Thumbnail displays should have an aspect ratio resembling a page or regular icons. Web pages, however, are often long documents that require scrolling. In Figure 5-8, we see what a web page's (http://www.cpsc.ucalgary.ca) thumbnail would look like if we included the entire document. This is actually a document about three 'window pages' long. Others can, of course be longer.

Figure 5-8. Complete page thumbnail.

One solution to this problem is to clip the page to a reasonable size. A possible strategy is to grab only the portion visible in the web browser window. Since the web browser window is usually displayed within a square-like shape, using only the portion visible on the screen the image will scale down to a square icon. Of course, there is no guarantee that the window is square. Additionally, the image may be 'corrupted' if it is partially covered by other windows.

Unified History uses this approach, with a slight modification to overcome these problems. Rather than take a 'snapshot' of the user's browser, it creates a hidden browser window of a specific width and height and captures that as a thumbnail. This hidden browser technique provides several advantages. First, the aspect ratio remains constant, so even if the user resizes their window to a different size, the thumbnails will all have the same dimensions. Second, because it is a hidden window, no windows overlap it and therefore are not included in the thumbnail image.

5.4.2 Timing

Another issue of importance is when to take the snapshot. Unlike images that consist of only one file to load, web pages contain multiple image files. When a web page is loaded, the text usually first appears, followed by slower-loading images. Taking a snapshot too soon means that there will be blank spaces in the image. We need to know when the page is completely rendered before the thumbnail can be taken.

Luckily, Internet Explorer provides the programmer with an event indicating when the entire page has been loaded completely. At this point, it is safe to take the snapshot. However, this assumes that the user has patiently waited for the page to finish loading before navigating to a new page. In practice, hasty users often click on a link before the page has finished loading. In these circumstances, we should take the thumbnail before the new page appears. Although there will be blank spaces in the thumbnail, it will accurately reflect what the user saw before navigating.

Unified History uses the following approach; if a page is completely loaded, it takes the snap shot. If the user navigates before the page is completely loaded, it takes the snap shot, even though the image will contain empty spaces. Either way, the image reflects how the user last saw the page.

5.4.3 Scaling

After obtaining the image of the web page, we need to scale down the image to its ‘thumbnail’ size. No matter which thumbnail size we choose, the scaling algorithm is important. Chapter 4 discussed the importance of text legibility in the thumbnail. If the scaling algorithm produces ‘jaggy’ text, the user loses much of the thumbnail’s information.

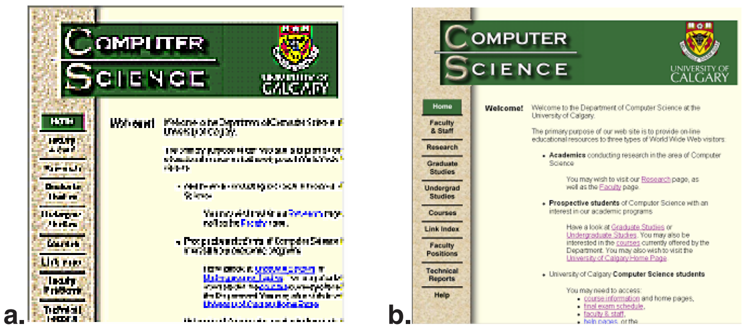


Figure 5-9. Comparing (a) row and column deleting algorithm with (b) box filter.

Unified History uses a ‘box filter’ to smooth as it scales the thumbnail. A box filter averages the colours surrounding a particular pixel. In Figure 5-9a, we see a thumbnail scaled using the standard ‘row and column deleting’ method, taken from Windows 98’s web page preview tool. The edges in the thumbnail are not smooth, making the text in event the large thumbnail difficult to read. In Figure 5-9b, we see the thumbnail scaled using the ‘box filter’ method. The edges are much smoother, making it much easier to read the text on page[9]. Of course, the box filter algorithm requires more computation than

[9] The ‘box filter’ graphics component was produced by Michael Boyle, and is available for download at http://www.cpsc.ucalgary.ca/grouplab

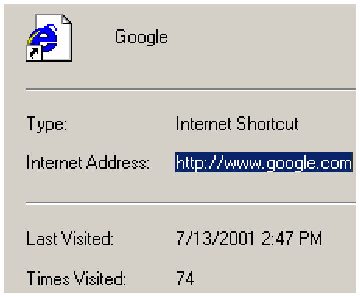
the row and column deleting algorithm. However, we do not believe the performance (speed) difference is noticeable to the user.

5.4.4 Saving/recalling

After scaling the thumbnail image, we must save it so that the history list can recall and display it. Fortunately, web pages provide a name that is guaranteed to be unique—its URL. Unified History saves all thumbnails into a single file, with each thumbnail's image information in a separate 'stream' of data. Each thumbnail image, stored at a size of 135x135 pixels, requires 55 kilobytes. This size is somewhat large, since the images are stored as bitmap data. To reduce the memory usage, the images could be saved in a compressed format such as JPEG. However, the compression/decompression requires more computation than a regular bitmap, which means they will load and save slower than regular bitmaps.

5.5 Tracking history

As shown throughout this thesis, Internet Explorer currently has its own history devices. Implementing the Unified History features was made easier because Microsoft's IE history data was connectible to end programmers. Thus, I did not have to implement data recording in order to know what URLs the user has visited. Had I been forced to implement my own history tracking, there would be no immediate history data available when the user installed Unified History. Thus, the user would not have any benefit from the software until it had recorded a reasonable amount of history data.



Google

Type: Internet Shortcut

Internet Address: http://www.google.com

Last Visited: 7/13/2001 2:47 PM

Times Visited: 74

Figure 5-10. Internet Explorer's History system keeps track of 'Times Visited'.

Using Microsoft's tracking system allowed Unified History to access the titles, URLs and last visited date of all pages the user had visited within the last 3 weeks[10]. However, an important part of IE's history information was not accessible— the hit count (how many times the user has been to each page). As seen at the bottom of Figure 10, Microsoft's own history system displays the hit count for pages ('Times Visited'). However, there is no interface for accessing this data in other programs.

5.5.1 Implementing visit count

I had two options for including hit count data.

1. Unified History track the hits number
2. Use other measures instead.

I chose option 2, for the same reasons I decided to use IE's history tracking. If I kept my own hit count record, there would not be any hit-count information until the program had accumulated enough data. Using a different but system supplied measure would bring immediate benefit even if it were only an estimate.

I used the Internet Cache to provide an approximation of hit count. Standard browsers include an Internet Cache that stores a copy of recently visited web pages on the user's computer. The cache speeds up web browsing by having the parts of a web page that have not changed recently loaded off the local computer rather than downloading it from the web site. Internet Explorer's cache is connectable, meaning that my program can query the cache for information about web pages. Specifically, I can access the count of how many times Internet Explorer has loaded the cache content for each web page. This provides a close approximation of the visit counts of pages. The only difference is that the cache contents are independent of the history record. While the history list saves page

[10] This is IE's default setting, which users can modify to their preference.

information for a period of three weeks, the cache discards its pages based on disk size. When the cache becomes too large (e.g., occupying more than 10% of the drive), Internet Explorer cleans out the oldest items. However, since both the history record and the cache clean out the oldest items, they will have similar sets of pages.

An even better method would be a combination of this measure and internal tracking in Unified History. If Unified History has recorded more hits than the cache, use Unified History's count. Otherwise, use the number from the cache. Following this strategy ensures Unified History does not lose any information that it previously had when the cache is cleared.

5.5.2 Suggestion for browser designers

Since Internet Explorer tracks visit count, it would helpful to end developers if this information was exposed. Of course, exposing program information may not agree with company competition strategy. However, since the history data and cache data is available, it is not clear why the visit counts are hidden.

5.5.3 Pruning history

Obtaining history is only half of the problem. The second half is deciding how much of it to keep. There are reasonable limits to how much memory one can devote to storing history. At some point, the less important items need to be removed. Since Unified History utilizes Internet Explorer's history data base, it is dependent on IE's system of pruning history. By default, IE only keeps the most recent 3 weeks of data. However, Unified History augments this by not allowing any dog-ears from being pruned out. The system could be improved even more, by not pruning high visit count items as well.

5.6 Implementing Unified History in an Explorer bar

I have built several history prototypes in the last two years. As part of my undergraduate honour's project, I designed a history list system in Visual Basic, where Figure 5-11 illustrates the final prototype. Its interface is somewhat similar to the current Unified History version. It includes a title search filter (labelled 'Name'), a visit-count filter (labelled 'Hits'), and a Recency Ordered Histoy list. Dog ear pages are marked by '*'. No thumbnails were used, only generic Internet icons. One of the shortcomings of this system is that it appeared as a separate window, rather than an integrated part of Internet Explorer. Thus, using it requires switching between windows, leading to the problems described in Chapter 2.

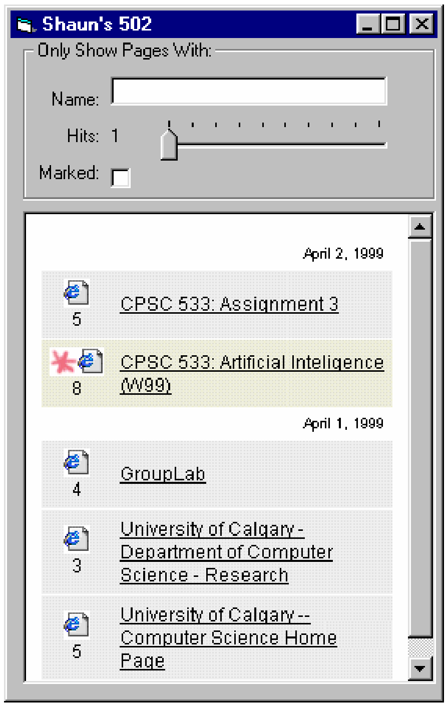

Shaun's 502
Only Show Pages With:
Name:
Hits: 1
Marked:
April 2, 1999
5 CPSC 533: Assignment 3
8 CPSC 533: Artificial Inteligence (W99)
April 1, 1999
4 GroupLab
3 University of Calgary - Department of Computer Science - Research
5 University of Calgary -- Computer Science Home Page

Figure 5-11. System implemented for undergraduate project.

An important goal was to build a history system that integrates directly into Internet Explorer, for I believe that few people would use it otherwise. This required learning Microsoft's COM (Component Object Model) and the Win32 Programming extensions for C++. COM programming is necessary to order to 'register' the program as an add-on for Internet Explorer, where Internet Explorer would spawn and communicate with my program. These concepts had a large learning curve, taking several months to approach the functionality similar to my early Visual Basic prototype. However, the pay off has been considerable. I have successfully implemented a design that is better than the previous prototypes in several ways. The system is much faster, and therefore more usable than the early prototype. Also, the system is able to connect to IE's history information, which would have been quite difficult from the Visual Basic prototype. Finally, I have assisted several other researchers from abroad who have needed similar functionality.

Conclusion

This chapter discussed some of the difficult challenges that arose in implementing Unified History. While these points do not relate directly to history list design, researchers and practitioners will likely encounter many of them. The following table summarizes these issues' problem, work around solution, and how browser developers could ease this problem for end developers.

Table 5-1. Technical problems and work arounds.

Problem	Solution	Recommendation to browser designers
Browser supports window rather than global history.	Use a server process to keep the global history.	Add end developer support for window to window communication.
There is no built-in communication between revisitation facilities.	Use a server process, to handle facility to facility communication.	Add end developer support for facility to facility communication (e.g., so the history list knows when a new navigation is made).
Frames pages are difficult to manage in history.	Compromise by treating initial frame set as an individual page, and each subsequent frame navigation as a new page.	Add support for frame sets, rather than only individual pages.
Web page thumbnail images require attention to aspect ratio, clipping, timing, scaling, and storing details.	Take a snapshot of the browser window (preferably a hidden window), scale using a box filter algorithm, and store the file using its URL as its index.	Add end developer support for web page developers at the Operating System level.
While history information is exposed by the operating system, visit count data is hidden to end developers.	Use the cache access count as an estimate for web page visit count.	Add end developer support for accessing the visit counts for web pages.

6. Conclusion

6.1 Goals achieved

At the end of Chapter 1, I outlined goals for this thesis. I will now repeat these goals and then describe the extent to which each of these goals has been met.

1. **There is a lack of integration between revisitation facilities**. I will produce a single integrated system for page revisitation. I will do this by unifying Back, History and Bookmarks into a single interface, displayed in a single window.

   I have developed the Unified History system, which demonstrates how Back, History and bookmarks can work together. Back/Forward are synchronized with the history list, so that the user can look at the history list to see where Back/Forward will lead. Bookmarks are just specially marked items in the history list, so that there is no longer a separate bookmark collection to organize.

2. **There are identified problems with the stack-based model of the Back/Forward buttons**. I will produce Back/Forward buttons that operate on a recency model. Previous researchers have found that this model reflects how users expect the buttons to work. Once these are implemented, I will verify through a user study that people can use these buttons at least as well as the current stack-based model buttons.

   I have implemented Back/Forward to use the Recency model. I was involved with a user study that found that users did not have a preference between the stack and recency model buttons (Appendix 1). This finding suggests that changing the behaviour of Back may be warranted, as the recency model allows users to see exactly where Back/Forward leads in a Recency-sorted history list.

3. **Recognizing particular pages in the history is difficult.** I will investigate the use of thumbnail images and analyze their effectiveness through a user study – comparing user's recognition of thumbnails vs. titles and URL addresses.

I have performed an extensive user study investigating how well people can recognize web pages by title, URL and thumbnail. One of the main findings was that people are better at recognizing exact page details when given the thumbnail than with title or URL. Thus, there are major gains in using thumbnail images in a revisitation system.

4. **Finding particular pages in the history list is difficult**. I will design the History list based on a single recency-ordered list, so that recently visited pages are easily accessed. Also, I will implement dynamic query filters to help the user isolate desired pages based on several criteria.

   Unified History's history list is recency-ordered, leveraging the recommendations by Tauscher and Greenberg (1997). Unified History also contains dynamic filters for finding pages by web site, page title, and number of visits. In particular, many Unified History users have remarked that the title filter has come in handy often.

5. **Bookmarks have several recognized problems.** By unifying the history list and bookmarks, I will produce a bookmark system where new bookmark items are made prominent and older bookmark items gradually migrate out of focus. The dynamic query filters will allow bookmarks to be searched by several criteria.

   Unified History contains Dog Ear pages, which are lightweight bookmarks that exist in the history list. Being in the history list, these Dog-Ear items can be searched like regular history items using dynamic filters for web site, page title and number of visits. Similarly, implicit bookmarks are created by inferring their importance, measured by visit counts.

6. **Many alternatives to the standard revisiting mechanisms require heavy use of screen real estate**. I will consider only designs that are constrained by how much screen real estate they use. In particular, the design will use roughly one quarter of the browser window width, comparable to the standard History bar provided in Internet Explorer.

   Unified History occupies the same amount of space as IE's regular history bar. Thus, it does not require its own screen like other alternative systems.

6.2 Contributions

This thesis has taken a very practical, rather than theoretical approach to web page revisiting. In particular, it has identified specific problems in today's revisitation facilities and suggested ways to amend these problems. This thesis also describes a user experiment that we believe is the first of its kind to test users' recognition of thumbnails, titles and URLs for pages they visited in real work context. Finally, this thesis has described some of the architectural and implementation issues that are necessary to produce an integrated revisitation system. The problems encountered and the work-arounds we used should be helpful for researchers planning to replicate our system or deviate into other novel revisitation system designs.

6.3 Future research

There are many areas that can be pursued in future research. First, a thorough usability study of the Unified History system is needed. This could include which features of Unified History are successful at mitigating the problems described in Chapter 1, and which features are inadequate. This study could also compare how user navigation habits change when using this system compared to the behaviour found by Tauscher and Greenberg's (1997) study.

The thresholds study highlighted many areas that can be pursued. Our finding that 30% of pages did not have titles certainly demands further research. We are doubtful that this percentage applies to general users, but have not found a clear reason that this occurred with our subject pool. Also, combinations of stimuli (titles with URL, titles with thumbnails, etc.) are important as it is likely that the users will see more than one cue, as they do in Unified History.

Finally, the Back/Forward buttons warrant more research as more and more interfaces are being migrated to the web. These buttons could be the central controls for a substantial percentage of computer interfaces. Thus, any improvements that can be made with them will have enormous impact on computing in general.

7. References

Abrams, D., Baecker, R. and Chignell, M. Information Archiving with Bookmarks: Personal Web Space Construction and Organization. In Proceedings of the ACM/SIGCHI Conference on Human Factors in Computing Systems (CHI'98), pages 18-23, 1998.

Ayers, E. and Stasko, J., Using Graphical History in Browsing the World Wide Web. In Proceedings of the Fourth International World Wide Web Conference, Dec 1995.

Catledge, L and Pitkow, J., Characterizing Browsing Strategies in the World Wide Web. In Proceedings of the Third International World Wide Web Conference, April 1995.

Cockburn, A. and Greenberg, S., Issues of Page Representation and Organisation in Web Browser's Revisitation Tools. In Proceedings of the Australian Conference on Human Computer Interaction (OZCHI'99), Wagga Wagga Australia, November 1999.

Cockburn, A. and Jones, S. 1996, Which way now? Analysing and easing inadequacies in WWW navigation', International Journal of Human-Computer Studies 45(1), 105–129

Cockburn, A., and McKenzie, B., What Do Web Users Do? An Empirical Analysis of Web Use. In International Journal of Human-Computer Studies (2000)

Cockburn, A., Greenberg, S., McKenzie, B., Smith, M., and Kaasten, S. Webview: A graphical aid for revisiting web pages. In Proceedings of the 1999 Computer Human Interaction Spacialist Interest Group of the Ergonomics Society of Australia (OzCHI'99). November 28-30 Wagga Wagga., pages 15-22, 1999.

Czerwinski, M., van Dantzich, M., Robertson, G., Hoffman, H. The Contribution of Thumbnail Image, Mouse-over Text and Spatial Location Memory to Web Page Retrieval in 3D. In Proceedings of Interact '99, pages 163-170, 1999.

Greenberg, S. (1993a). The Computer User as Toolsmith: The Use, Reuse, and Organization of Computer-based Tools. Cambridge series on human-computer interaction. Cambridge University Press.

Greenberg, S. (1993b). Supporting command reuse: mechanisms for reuse. Int. J. Man-Machine Studies, 39, 391-425.

Greenberg, S. (1993c). Supporting command reuse: empirical foundations and principles. Int. J. Man-Machine Studies, 39, 353-390.

Greenberg, S. and Cockburn, A., Getting Back to Back: Alternate Behaviors for a Web Browser's Back Button. In Proceedings of the Fifth Conference on Human Factors and the Web, June 1999.

Greenberg, S., Ho, G., Kaasten, S. (2000). Contrasting Stack-based and Recency-Based Back Buttons on Web Browsers. Report 2000-666-18. University of Calgary, Calgary, Alberta, Canada. August.

Greenberg, S. and Witten, I. (1988). Directing the user interface: How people use command-based systems. In *Proceedings of the 3rd IFAC Conference on Man-Machine System*s, Oulu, Finland.

Newfield, D., Sethi, B., Ryall, K. (1998). Scratchpad: Mechanisms for Better Navigation in Directed Web Searching. University of Virginia, Charlottesville, Virginia.

Pitkow, J., 'Gvu's www user surveys', WWW page: http://www.cc.gatech.edu/gvu/user_surveys/survey-04-1998/, 1998.

Shneiderman, B., Williamson, C. and Ahlberg, C. Dynamic Queries: Database Searching by Direct Manipulation. *Proc ACM CHI*, 1992

Tauscher, L. and Greenberg, S., How People Revisit Web Pages: Empirical Findings and Implications for the Design of History Systems. In International Journal of Human-Computer Studies, pages 47(1), 97-138, 1997.

Worlds, R., Robertson, G., Czerwinski, M., Larson, K., Robbin, D., Thiel, D., Dantzich. M. Data Mountain: Using Spatial Memory for Document Management Information Proceedings of the ACM Symposium on User Interface Software and Technology 1998 p.153-162.

8. Appendix 1: Back Button Study

Contrasting Stack-Based and Recency-Based Back Buttons on Web Browsers

Saul Greenberg[1], Geoffrey Ho[2] and Shaun Kaasten[1]

[1]Department of Computer Science and [2]Department of Psychology
University of Calgary
Calgary, Alberta, Canada T2N 1N4
+1 403 220 6087
saul@cpsc.ucalgary.ca

Abstract

People frequently use the ubiquitous Back button found in most Web browsers to return to recently visited pages. Because all commercial browsers implement Back as a stack, previously visited branches of the tree are pruned. While this means that people can quickly navigate back up the tree, previously seen pages on alternate child branches are no longer reachable through Back. An alternate method implements Back on a recency model. Here, all visited pages are placed on a recency-ordered list with duplicates removed, which means that all previously seen pages are now reachable via Back. Because advantages and trade-offs exist in both methods, we performed a study that contrasted how people used stack *vs* recency-based Back. Surprisingly to us, several of our results were contrary to our expectations. First, people have a poor model of *both* stack and recency-Back. Second, people have difficulty predicting what pages will appear as they click Back. Instead, they employ a 'click until recognize' strategy, where they simply click Back until they recognize the desired page. Third, people show no strong preference of recency *vs.* stack-Back. Consequently, we advocate replacing stack-Back with recency-Back only if other browser design considerations warrant it.

Key words: History system, browser design.

1 Introduction

A person's ability to find and navigate effectively to new information and to new web sites is extremely important, and this has driven many researchers to understand both how people navigate within the Web, and how Web sites should be designed. Equally important is a person's ability to return to pages he or she has already seen: page revisitation is a regular and surprisingly strong navigational occurrence. An early study by Tauscher and Greenberg [6] found that around 60% of all pages an individual visits are to pages they have visited previously. A later replication of this study by others found an even higher revisitation rate of around 80% [1].

Given this revisitation statistic, we believe that Web browsers should go to great lengths to support effective page revisitation. Indeed, most browsers do provide revisitation support through various mechanisms: the Back and Forward buttons, history lists, bookmark facilities, and even site maps that graph the pages that a person has visited [2][6].

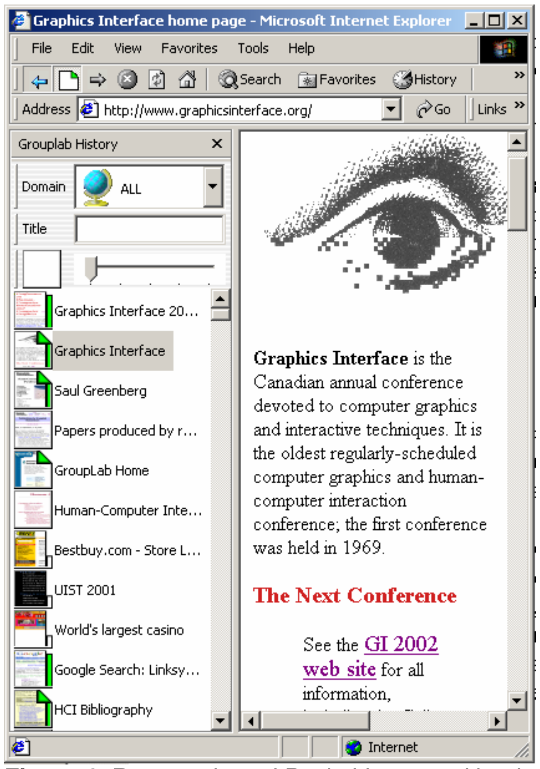
Of these revisitation mechanisms, it is the Back button whose use predominates: Tauscher and Greenberg [6] discovered that pressing the Back button comprised over 30% of all navigational acts. In contrast, other revisitation facilities are used infrequently e.g., <3% for bookmarks, and <1% for history systems.

Greenberg and Cockburn [4] detailed several reasons explaining Back's popularity.

1. Back allows people to rapidly return to very recently visited pages, which comprise the majority of page revisits. This is important, as Tauscher and Greenberg [6] found that there is a 43% chance that the next URL visited will match a member of a set containing the 10 previous visits. Because 60% of all pages are revisits, this means that $43 \div 60 = 72\%$ of all revisited pages are within 1–10 Back clicks.
2. Back requires little effort as a person merely clicks on it until the page is reached.
3. People are willing to keep Back on permanent display because it is visually compact.
4. People can use Back successfully even when they have a naïve understanding of the way it works [2].

Back's popularity as a revisitation tool means that it deserves special attention. Somewhat surprising to us is the wide-spread—and unchallenged—acceptance of the stack-based navigation model underlying Back and Forward in virtually all commercial browsers. If we can improve this behaviour even slightly, millions will feel the added benefit.

Consequently, our focus in this paper is to re-examine the usability of the way existing Back and Forward buttons work on a *stack*, and to compare it to an alternative button based on *recency*. We look at recency for several reasons (see Section 2.1). First, previous



Graphics Interface is the Canadian annual conference devoted to computer graphics and interactive techniques. It is the oldest regularly-scheduled computer graphics and human-computer interaction conference; the first conference was held in 1969.

The Next Conference

See the GI 2002 web site for all information,

Figure 1. Recency-based Back, history and bookmarks

research suggests that recency is closer to how people think Back actually works [2]. Second, the stack-Back loses pages, while recency-Back does not.

We also have a personal motivation. We have built a novel revisitation system that integrates Back, History and Bookmarks [5]. It represents pages on a sidebar as thumbnails and titles ordered by recency (Figure 1). Bookmarks are included in this list as specially marked 'dogeared' pages. Using dynamic queries, people can rapidly filter the list to show only frequently-visited pages, dogeared pages, pages within a particular domain, or pages whose title contains a particular string. Of particular relevance to this paper is that the Back and Forward buttons are also based on recency: they simply go up and down the list. While there are obvious advantages of making Back work directly on this visible list, we had no idea if a recency-based Back button would be acceptable to end users.

In this paper, we describe and contrast stack-based *vs.* recency-based Back button behavior. After summarizing how these behaviors work (Section 2), we introduce our study where we investigate how well people understand and use these two buttons (Section 3). We then present and discuss our results (Section 4), and we close by pointing out implications of our work to the design of web browsers.

2 Stack *vs.* Recency-based Back Buttons

This section summarizes two different behaviors for the Back and Forward buttons: the stack-based behavior found in today's web browsers, and a recency-based behavior proposed and implemented by Greenberg and Cockburn [4]. We will illustrate these two behaviors by showing how people navigate through the small page structure shown in Figure 2.

We use the notation $x \rightarrow y$ where '$\rightarrow$' means that the person has selected or typed a link on page x to go to page y. Similarly, in $y \Leftarrow x$, the '$\Leftarrow$' means backtrack from page y to page x via the Back button. We also define *hub and spoke* navigation as an action where people follow links from one parent (the hub) to two or more children (the spokes). For example, when navigating $b \rightarrow c \Leftarrow b \rightarrow h \Leftarrow b$ in Figure 2, b acts as a hub while c and h are spokes. Of course, c could also act as a hub page if the user navigates a similar pattern to two or more of c's children. This hub and spoke behavior deserves special attention because it is a common navigational act [6] and because it results in page pruning by stack-Back, as described below.

2.1 Stack

Description. The stack algorithm underlying a conventional Back button has three different types of operations.

1. Clicking or typing links adds a page to the top of the stack.
2. Clicking Back and Forward moves the stack pointer down and up the stack respectively, displaying the page at that stack location. The actual stack contents are not altered when navigating with these buttons.
3. When the user is at any position on the stack other than the top and selects a link on a web page, all entries on the stack above the current position are popped (or pruned) off the stack before the new page is added. This is critical, as pages popped off the stack can no longer be revisited using the Back and Forward buttons.

To illustrate these steps and how the stack behavior affects what people see, let us say a person follows the page links in Figure 2 from pages a through d in order, then presses Back twice to return to page b, and then selects a new link on page b to page h. Figure 3a shows the stack after a person navigates $a \rightarrow b \rightarrow c \rightarrow d$, where all pages were pushed onto the stack's top. In Figure 3b, we see that the two clicks of the Back button ($d \Leftarrow c \Leftarrow b$) moves the stack pointer down the stack to b. Navigating from $b \rightarrow h$ pops pages c and d off the stack (Figure 3c), and then adds page h to its top (Figure 3d). Thus pages c and d are no longer reachable through the stack-based Back button.

Advantages and Disadvantages. The consequence of using a stack algorithm is that it automatically prunes navigational branches when people use Back followed by link selection. This approach has some merit: after exploring a branch and selecting a new path of interest the user may no longer need the previous branch of exploration. Because these pages are gone, a sequence of Back clicks will always move one 'up' the page hierarchy, making it easy to return to parent hub pages.

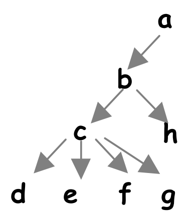
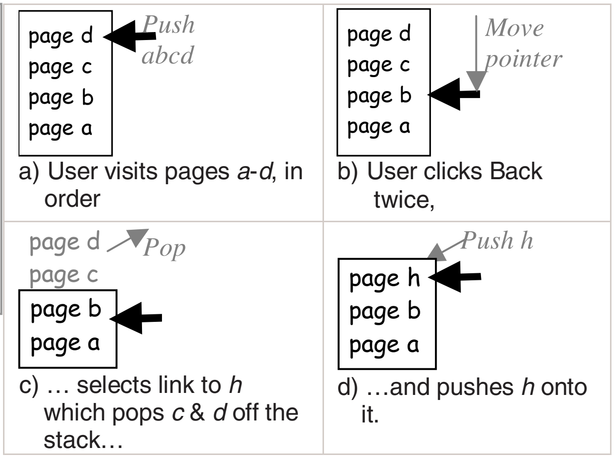


Figure 2. Example page structure.

A counter-argument is that there are many cases where people *do* want to return to pages seen on a previously visited branch. For example, in the navigational trace described in Figures 2 and 3, the spoke pages visited on the branches below hub page *b* disappeared as soon as another spoke of *b* (page *h)* was selected. If a person wanted to go back to spoke page *d* from page *h* (perhaps because they needed to review the information on page *d*), they could no longer do it via stack-based Back as page *d* has been pruned.

Still, we could argue that the Back button isn't really required for this case, because the person can first use Back to go from $h \Leftarrow b$, and then use the normal links on *b* to re-navigate $b \rightarrow c \rightarrow d$. While reasonable for short pages with few links and simple navigational paths, this could be onerous for more complex situations. First, many (badly designed) web pages now override the coloring of previously selected links, which makes them indistinguishable from unvisited ones and thus harder to find. Second, some pages are long and complex: recalling and finding the correct link within the page adds the extra burden of scrolling and searching. Third, finding the correct spot to re-click on image maps may be challenging. Finally, if the person navigated a complex path to a particular page, they may find it difficult to retrace that path later on.

Stack-Back has another problem. Current systems do a poor job of communicating stack's tree-pruning behavior to its users [2], and most people actually believe that Back just returns sequentially to one's previously seen pages (this incorrect view is validated further in Section 4.1). This discord between how Back works and how people think it works is no surprise. The labels Back and Forward imply linearity, rather than of a tree. There are few cues at the interface to help users distinguish between the underlying semantics of page display using link selection (how new pages are added and how the stack is popped) *vs.* the semantics of moving within the stack using the Back and Forward buttons. Consequently, users sometimes wonder why pages are seemingly 'lost' when using Back.


page d
page c
page b
page a
Push abcd
a) User visits pages a-d, in order
page d
page c
page b
page a
Move pointer
b) User clicks Back twice,
page d
page c
page b
page a
Pop
c) … selects link to h which pops c & d off the stack…
page h
page b
page a
Push h
d) …and pushes h onto it.
Figure 3. An example navigational trace and its effect on the stack. Note that pages *c* and *d* are popped off the stack.

2.2 Recency

Description. Perhaps the greatest disadvantage of stack-Back is that it cannot guarantee that previously visited pages are reachable by successive Back clicks. As an alternative, we could provide a complete history of all visited pages by having Back and Forward move a person through a recency-ordered list, where the buttons simply navigate through the pages in reverse order to how they were seen. Surprisingly, the design of recency-Back is not as simple as might be expected. Greenberg and Cockburn [4] explored several models of Back based on variants of a recency-ordered history list: here we describe only the final one that they advocate: recency with duplicates removed and a temporal ordering enhancement.

We begin with a side discussion: the management of duplicate entries in the history list. When a person sees a page more than once, the system would record them as duplicate entries on the recency list. The advantage is that successive Back clicks would go through a literal representation of the order of pages that the user has seen. The disadvantage of retaining duplicates is that the list (and thus the number of Back clicks) could become unnecessarily long and repetitious. Instead, Greenberg and Cockburn [4] suggest pruning duplicate pages by keeping only a single copy of it in its most recent position on the list: this keeps recently revisited pages near the top and thus quickly reachable through Back. Tauscher and Greenberg [6] analyzed this approach, and found that substantially fewer Back presses would be required to return to a desired page when duplicates are pruned.

The algorithm for maintaining a true temporal recency list with duplicates removed is described below.

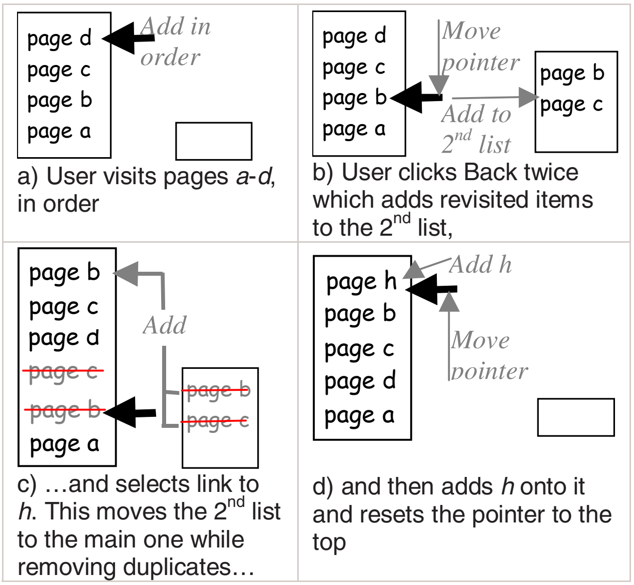
1. As with stack, clicking or typing links displays a page and adds it to the top of a recency list.
2. Similarly, clicking Back and Forward moves a pointer down and up the recency list respectively, where the pointed at page is rendered in the browser. However, the page seen before the button is clicked is also added to the top of a second list.
3. When the user is at any position on the list other than the top and selects a link on a web page, the contents of the second list are moved to the top of the recency list. Any duplicate entries below the current ones are then removed. Finally, the new page is added to the top of the list and rendered in the browser.

The function of the second list introduced in Step 2 is to track the order of pages seen as a person navigates the recency list using Back and Forward. We need this list in Step 3 for reordering the primary list to its true temporal order after a new link is selected. This scheme matches the sequence of page as the user saw them (excluding duplicates), and works over any number and combination of link selections and Back and Forward actions.

Figure 4 illustrates how this algorithm works using the same set of pages and the navigation example of Figures 2+3. As before, visits to the pages *a*→*b*→*c*→*d* produces the main list {*d,c,b,a*} (Figure 4a). Going from *d*⇐*c*⇐*b* creates a second list {b,c} (Figure 4b). As soon as the person selects the new link *b*→*h, b* and then *c* are added to the main list and any duplicates of *b* and *c* further down are removed (Figure 4c). Finally, the new page *h* is added, giving {*h,b,c,d,a*} which is the correct temporal sequence of pages (with duplicates removed) that the user has just seen (Figure 4d).

Advantages and disadvantages. Recency has several potential advantages. First, the list of previously visited pages is complete because no pages are popped off the list. Therefore users are guaranteed to be able to revisit pages already encountered during their browsing session by using the Back button. Second, because the underlying recency list can grow indefinitely, it is feasible for Back to work *between* sessions i.e., successive browser invocations and login sessions. Third, the temporal reordering algorithm means that users always see a temporally correct retracing of their page path using Back, which likely matches how people perceive Back to actually work [2].

Yet one disadvantage of recency-Back arises if a user's goal is to navigate back up the tree to a parent hub rather than to a previously seen spoke page.



a) User visits pages *a-d*, in order

b) User clicks Back twice which adds revisited items to the 2nd list,

c) …and selects link to *h*. This moves the 2nd list to the main one while removing duplicates…

d) and then adds *h* onto it and resets the pointer to the top

Figure 4 The same navigational trace using a recency list.

Superfluous spoke pages are now interposed as recency does not prune those spoke pages visited on a different branch.

3 The Study

Is recency-Back a viable replacement for stack-Back? If it proves 'better', then browsers implementers should replace stack-Back with recency-Back. If it proves no better or worse, then implementers have the design option of using recency-Back if they can justify it e.g., as with the integrated revisitation system shown in Figure 1. If it proves worse, then we know that stack-Back is the preferred strategy.

To answer these questions, we designed a study that examined people's mental model of stack. We then explored how well a recency *vs.* stack-based Back matched peoples' expectations of how it worked. We did this by asking people to predict the pages that would appear as they navigated via Back. We also asked people which button they preferred. We had several expectations, framed as hypotheses.

Hypothesis 1. Users have a poor mental model of stack-Back. We expect this as it echoes an earlier result [2].

Hypothesis 2. When revisiting pages over different navigational paths, users are better predictors of what pages will appear when using recency-Back *vs.* stack-Back. We expect this because people believe Back navigates through the pages as they were seen [2].

Hypothesis 3. After using both a stack-based and a recency-based Back button for a similar set of tasks, people will prefer recency. We expect this because recency better fits peoples' reported mental models of Back [2].

3.1 Participants and expertise

Thirty volunteers participated in this study. All had some level of post-graduate education. Answers to a pre-test questionnaire indicated a mixed but generally web-savvy group. For web usage: 20 participants claimed to use a browser almost every day; 5 stated their use as ranging from once every few days to once a week; while the remaining 5 reported low Web use. Participants described themselves as: 4 being skilled experts, 13 having good (but not expert) skills, 7 having basic skills, and the remaining 6 having beginner-level skills. All participants stated that either Netscape Navigator or Microsoft Internet Explorer was their preferred browser.

3.2 Materials

Participants used Microsoft's Internet Explorer version 5.0 running within Windows 98 on a modern PC with a 1024x768 24-bit color display. All pages used for the study were stored on the local computer. In essence, this configuration meant that navigations and resulting page displays were uniformly rapid.

We added two non-standard software systems to the browser. First was a (visually identical) Back button that used the recency with duplicates removed/temporal ordering enhancement algorithm [4], as described in Section 2.2. Second was an artificial search bar used for one of the tasks: while it resembled a browser's typical search bar result, it actually contained a pre-defined set of static links.

We used several local web sites. We created these by importing/modifying a few popular commercial sites.

3.3 Method

The entire procedure listed below required approximately one hour of participant's time.

Stage 1: Initial mental model. Hypothesis 1 claims that people have a poor mental model of Back. Our strategy was to confront people with how they thought Back worked with how stack-Back actually worked. First, we asked them to articulate their mental model of the (stack-based) Back button they normally use (Figure 5, question 1). Second, we gave them a simple web site—a book table of contents containing links to several chapters—and had them navigate the hub and spoke pattern $a \rightarrow b \Leftarrow a \rightarrow c$. Participants did not see this arcane notation; they were told to go from the table of contents to Chapter 1, then Back to the contents, and then to Chapter 2. Third, we then asked participants how many times they would have to click the Back button from their current page *c* to go back to *b* i.e., from Chapter 2 Back to Chapter 1 (Figure 5, question 2). Fourth, we then told participants to try to return to *b* by actually using the standard stack-Back button. Finally, they were asked questions 3-5 in Figure 5 about how the observed stack-Back behavior matched the answer they previously gave in Question 1.

1. Describe how the back button works, and how the Back button internally manages and stores the pages you visit.

Participants were then asked to navigate through three pages comprising a simple hub and spoke system $a \rightarrow b \Leftarrow a \rightarrow c$.

2. How many times would you have to click Back to return to <page *b*>?

They were then told to try to return to a via Back.

3. Were there any problems?
4. Did this match your model of the back button in question 1?
5. Is the version of the Back button you are using is the same as the one supplied with your normal web browser?

Figure 5. Questionnaire excerpt concerning people's mental model of the Back Button.

Stage 2: Navigation and prediction tasks. Hypothesis 2 suggests that people can better predict what pages they will see when using recency-Back vs. stack-Back. To check this claim, we randomly assigned participants to one of two groups, where each group saw either the stack or recency-Back button first. We gave each participant five different and increasingly complex tasks. Instructions for all tasks are summarized below.

1. We reminded participants to think aloud as they worked.
2. From a home page, we had participants navigate through a series of links to a destination page. Participants had to scan each page they saw in order to find and choose the correct next link.

3a. We asked participants to return to a particular previously visited target page using only the Back button.

3b. Before each and every Back click in step 3a, participants had to predict what page they expected to see. They described the expected page, clicked Back, and then stated if their prediction was correct.

The five tasks stepped through different navigational sequences, corresponding to those illustrated in Figures 6–9 and described below.

Short linear sequence (Figure 6). The participant navigates from page *a* (the home page of the Discover Alberta web site we used) through two intermediate pages *b* and *c* to reach the destination page *d* (a page describing hostels in Edmonton) i.e., $a \rightarrow b \rightarrow c \rightarrow d$; this is illustrated by the straight arrows in the Figure. The participant then uses Back to return to page *a,* making predictions before each click. Correct predictions are denoted in Figure 6 as P1 to P3 (for predictions 1 to 3). Because there are no branches, both recency and stack behave identically i.e., a is returned to by $d \Leftarrow c \Leftarrow b \Leftarrow a$ (the curved arrows in the Figure).

Long linear sequence (Figure 7). This task is similar to the one above, except that more intermediary pages are involved. Reaching destination page i requires $a \rightarrow b \rightarrow c \rightarrow d \rightarrow e \rightarrow f \rightarrow g \rightarrow h \rightarrow i$. Returning to a using both recency and stack Back is by $i \Leftarrow h \Leftarrow g \Leftarrow f \Leftarrow e \Leftarrow d \Leftarrow c \Leftarrow b \Leftarrow a$

Hub and spoke with return to hub (Figure 8). Reaching destination page h after visiting all the children of d requires $a \rightarrow b \rightarrow c \rightarrow d \rightarrow e \Leftarrow d \rightarrow f \Leftarrow d \rightarrow g \Leftarrow d \rightarrow h$. Returning to revisit target hub a using the stack Back just goes up the hierarchy by $h \Leftarrow d \Leftarrow c \Leftarrow b \Leftarrow a$ (4 Back clicks). Recency Back takes 7 clicks, as all d's children are seen again $h \Leftarrow d \Leftarrow g \Leftarrow f \Leftarrow e \Leftarrow c \Leftarrow b \Leftarrow a$. These paths and corresponding predictions are denoted in the figure as PR1-7 for recency predictions, and PS1-4 for stack predictions.

Hub and spoke with return to spoke, then hub (Figure 8). This is almost identical to the task above (although using a different set of pages). The only difference is that participants are first asked to revisit the child spoke page f of hub d, and then the root page a. Note that spoke page f is not reachable via stack as it is pruned. The revisitation sequence of both recency and stack Back are identical to the hub and spoke task above.

Search bar (Figure 9). This complex task simulates a user navigating through several sites by using the results of a search presented in a search bar. The order of navigation by the participant (where sb denotes the search bar) is: $sb \rightarrow b \rightarrow c \Leftarrow b \rightarrow d$; $sb \rightarrow e$; $sb \rightarrow f \rightarrow g \rightarrow h \rightarrow i$; $sb \rightarrow j \rightarrow k$. The participant is then asked to return to target page f, and then to c. The path of stack Back is $k \Leftarrow j \Leftarrow i \Leftarrow h \Leftarrow g \Leftarrow f \Leftarrow e \Leftarrow d \Leftarrow b$ (8 Back clicks). Note that because stack Back pruned spoke page c when the participant went back to b, target c is not reachable. Recency Back takes 9 clicks, and includes target page c $k \Leftarrow j \Leftarrow i \Leftarrow h \Leftarrow g \Leftarrow f \Leftarrow e \Leftarrow d \Leftarrow b \Leftarrow c$.

Stage 3: Subjective preferences. Hypothesis 3 claims that people would prefer recency-Back. To check this, we had participants redo the entire set of tasks performed with the first type of Back button with the other type of button. Through a post-test questionnaire, we then asked them to comment on each Back button type and which they preferred.

4 Results

We first describe our participants' mental model of stack-Back. We then report both the prediction and preference data of only the first 15 participants for reasons that will become apparent in the subsequent discussion. Afterwards, we present the preferences of the remaining 15 participants.

4.1 Mental Model of the Stack-based Back button.

When asked to describe how their conventional stack-Back button worked (Figure 5 question 1), the description of all but two participants indicated an incorrect or incomplete mental model. Most participants simply said that Back just returns to all previously viewed pages. Some were more explicit (but still incorrect), where they said that pages are stored and displayed as a list of all pages in the order seen. A few other answers hinted that participants were aware of the stack-Back pruning behavior, but even then they had an incorrect view as to when and why this happened. To quote several participants, Back:

- goes to previous page, but sometimes you can't…I think it goes back to [a] different user;
- takes you back to previous pages in your navigation path…does seem to fail;
- takes you back to last few pages you visited but after a few clicks it takes you to the main pages (only).

We then confronted participants with how the stack-based Back button actually worked. As mentioned previously, after they navigated a simple hub and spoke pattern, they were asked to predict how many Back clicks were required to return from the second spoke to the first spoke. The correct answer is that it is not possible, as stack-Back will have pruned it off the list. However, only 2 of the 30 participants correctly answered this question (the same two that knew about the stack); the 28 others incorrectly predicted two Back button clicks.

We then asked people to try and navigate to that page via Back, only to find that they could not. Most, but not all, admitted that it did not match their mental model as stated when replying to Question 1 (Figure 5). A few said that in hindsight it did, but not for the correct reason. For example, one person said "there are often errors…it often doesn't work at all". Two others said that "some sites just do this". Also interesting is that about half of all participants thought that the Back button they just used did *not* behave the same as the one they normally used, even though it did!

What is clear is that people have a poor mental model of stack-Back i.e., Hypothesis 1 is supported. Most people were not able to articulate how stack-Back really worked. They could not predict its pruning behavior in even a very simple hub-and-spoke example. They were either surprised when the system did not work as predicted, or they had an arcane rationale for why it behaved the way it did. This result accords with Cockburn and Jones' [2] study involving ten computer scientists. As their study was done in the fairly early days of web browsers, we could have argued that today's web users are more browser-literate. This replication of their findings clearly demonstrates this is not the case: users find the exact behavior of the Back button as inscrutable now as it was then.

4.2 Predicting pages returned to by Back.

We asked the first 15 participants to predict what page they would see before they pressed Back as they tried to return to the revisit target. We will explain shortly why we did not ask the remaining 15 participants to do this prediction task. Eight participants began with recency-Back, while the remaining seven used stack-Back. The results of these 15 are summarized below.

Short linear sequence. Each participant made three predictions P1, P2 and P3 for this sequence. With the exception of a single prediction error at P1 by one person, all successfully predicted what page would appear. The graph in Figure 6 illustrates this: the prediction number is on the X-axis, and the total number of errors made by the participants is on the Y-axis.

Long linear sequence. Each participant made eight predictions P1 through P8 as they navigated back to the revisit target (Figure 7). Unlike the short linear sequence, many errors were made, as shown in Figure 7's graph. While participants were more or less accurate for the first two predictions, errors became frequent.

Hub and spoke with return to hub. In this task, the target page was a hub up the hierarchy (the root page *a*). Four of the seven participants using stack-Back erred only in their second prediction PS2 (Figure 8, graphs at top). That is, there seemed to be some confusion as to where Back would take them after reaching the hub page *d*. In contrast, the eight participants using recency-Back made many prediction errors. As with stack, they were uncertain of what would happen after first reaching hub page *d* (PR2) and then again after seeing the second child (PR3)—most thought it would return to hub page *d* again. Only one person made an error on PR4, likely because they now realized they would see all children in order. However, many then expected to see hub page *d* again on PR5 rather than the hub's parent *c*.

Hub and spoke with intervening child. The only difference between this and the previous task is that participants were asked to revisit the spoke page *f* before revisiting hub page *a*. We wanted to see if predictions were better or worse if they were looking for a child spoke page instead of a page up the hierarchical path. Comparing errors with the previous hub and spoke navigation with stack-based Back (where target page *f* is unreachable), we see two additional errors on PS3 (Figure 8, graphs at bottom). We surmise that participants thought they would see the other spoke pages at these points. Recency-Back seems to have somewhat fewer errors when going through the first few children (PS2 and PS3), although the error rate is somewhat higher afterwards. As before, many expected to see hub page *d* after each spoke was revisited.

Search bar. The error rate for predictions on this complex revisitation task was very high for both conditions (Figure 9). The only specific data point worth special mention is Prediction 9 (PS9) for the stack: all predicted another page would be seen. This is wrong: no more pages could be revisited at this point since this was the top of the stack.

Preferences. After completing all tasks with one Back button type, participants repeated them with the other button. While the web sit used was different, the navigational structure was identical. We then asked them which button they preferred. Nine favoring stack, four recency, and two were undecided. Comments by participants suggested that predictions were easier to make with the stack model as it skipped sub-pages.

Think-aloud. During all tasks, we observed participants as they formed their predictions and thought-aloud about how they were making them. What was immediately obvious after running just a handful of participants was that predicting the next page often required a great deal of cognitive work as well as time. We saw participants try to mentally reconstruct where they had been: they would search the current page for clues as to what its parent could be while trying to recall what they had seen. They seemed to fare better on pages that had a logical hierarchical or path structure, and less so on pages whose structure was somewhat more arbitrary.

4.3 Discussion Part 1

On the surface (and without looking at statistical significance) Hypotheses 2 and 3 are rejected: stack-Back seems better than recency. Participants' error rate for predictions appears lower, and a majority of participants (9:4, 2 neutral) preferred stack to recency. This poor overall performance not only supports Hypothesis 1 but also suggests that people have a poor mental model of how a recency-Back button works.

Yet something is wrong with this story. In the think-aloud observations for both conditions, we saw participants expend a great deal of time and effort when making predictions; the overall error rate was also very high. This does not accord to how we see people using Back in everyday use: navigating with Back is done without much apparent thought, and people backtrack through successive pages very quickly and successfully.

This discordance led us to rethink our rationale for Hypothesis 2. We initially thought that people could anticipate or predict from their mental model what specific pages would appear when clicking Back, and they somehow did this in their everyday use of browsers. From our observations, this is clearly incorrect. Instead, our findings suggest an alternate Hypothesis 2:

Hypothesis 2 (alternative). When revisiting pages over different navigational paths, users are poor predictors of what pages will appear when using either recency-Back or stack-Back. Instead, people use a 'click until recognize' strategy, where they have a vague expectation that the searched-for page will appear sometimes, and they simply click the button until they recognize the desired page.

This alternative Hypothesis 2 also raises doubts about our previous rejection of Hypothesis 3, for the prediction task does not reflect how people actually used Back. Forcing them to make predictions almost certainly interfered with their goal of returning to the target page. This may have led to some preference bias for stack-Back, which is likely easier to predict because one just has to reconstruct how one moves up the hierarchy.

Participants' difficulties for page prediction were so striking that we felt no need to statistically analyze our data further, or to have our remaining 15 participants suffer through the prediction task. Rather, we altered the study on the fly to re-evaluate Hypothesis 3 as described below.

4.4 Comparing Stack *vs* Recency-Back Without Predictions

We re-examined Hypothesis 3 by seeing how participants would rate their preferences to the two Back buttons if they did not have to make predictions. We continued the study with the next fifteen participants exactly as before, except that we omitted step 3b in stage 2 of the method outlined in Section 3.3.

Unlike the previous 15 participants, we saw people quickly and effortlessly returned to the target page (if it was reachable) using both Back buttons. Where previous participants favored stack, this new set of participants showed no strong leaning for one button type over another. Eight preferred recency, six preferred stack, and one was undecided. People who preferred recency commented:

- go through the actual order more than not;
- pages come back sequentially as they should;
- more predictable: goes through the actual order;
- doesn't feel like more clicking;
- stack missed a whole bunch of pages;
- more intuitive… liked [that it had] no duplicates.

People who preferred stack commented:

- more used to it;
- recency produced extra clicks;
- doesn't take you back to sub-pages.

4.5 Discussion Part 2

Hypothesis 3 is still rejected, as people did not prefer recency over stack-Back. However, our new results suggest the following alternate hypothesis:

Hypothesis 3 (alternative). After using both a stack-based and a recency-based Back button for a similar set of tasks, people will not prefer one type of button over the other.

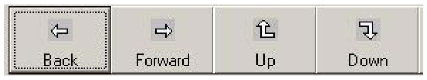
This change in preference as well as the ease which participants returned to target pages also re-enforces our conviction stated in the alternative Hypothesis 2. That is, people do not have an exact model of what pages they expect to see as they use Back, and that they use a 'click until recognize' strategy instead.

One point deserves further elaboration. A recurring comment made in regards to the recency model was the inability to recover 'hub' pages: participants expected to see the hub page after each visit to a child. Our recency with duplicates removed algorithm showed the hub page once, but the perception of the users was that it did not. Instead, they expected a pure sequential model. Does this suggest that hub pages should be duplicated rather than only shown once? We think not. With more extended use of recency Back, users may realize that the hub pages are accessible and may find the duplication of pages unnecessary. We also believe that seeing hub pages several times will introduce a different type of confusion i.e. that Back is merely cycling through the same sequence of pages. Still, more testing is needed before drawing a final recommendation of how duplicates are handled.

5 Implications to Browser Designers

At first glance, there is no compelling reason to change the current stack-based Back button to a recency-based one. People seem comfortable with stack-Back, even though they have a poor model of it. This is because their 'click until recognize' strategy does not require an accurate model of its behavior. As well, people seem somewhat unconcerned about the mysterious disappearance of pruned pages, blaming it on the vagrancies of computers. Importantly, there is no overwhelming preference by our participants of recency-Back over stack. While we could still argue that recency is better than stack because no pages are lost, we cannot make a compelling argument that the familiar stack-Back idiom should be replaced in conventional browsers.

However, the ambivalence between recency *vs.* stack means that new designs *can* include recency with no penalty. One possibility, which we have implemented in a prototype web browser, is to include both the stack and recency-based buttons. We relabel stack-Back and Forward as Up and Down, as this more accurately reflects the semantics of moving up and down the navigational hierarchy that is a side product of stack-pruning. Back and Forward are now recency-based, as they reflect the semantics of moving backwards and forward on the recency-ordered history list. Indeed, similar buttons are now found on several non-web browser products e.g., Microsoft's file explorer, and the MSDN document browser. Nonetheless, we recommend caution with these new buttons. Because users have a fuzzy notion of how stack and recency behave, the differences between these buttons may be unclear to them. As well, it adds complexity: yet another decision must be made as to which revisitation method should be chosen.



Perhaps a more compelling reason for using a recency-based Back button is to remove the differences between Back and the other revisitation systems available on web browsers. As previously described and as illustrated in Figure 1, our new revisitation system integrates Back, history, and bookmarks by unifying them to operate over a single recency-based list [4]. Back and Forward simply become shortcuts for navigating the history / bookmark list item by item. If the history list is visible, then items are highlighted as the user selects Back e.g., we see in Figure 1 that the 2nd item is marked, which means the person has just pressed Back once. This visually exposes and re-enforces how Back works. Our study suggests that this replacement of stack-Back with recency-Back can be done with no penalty.

6 Summary

Even though Back is probably the most highly-used interface widget in existence today, there are (to our knowledge) no other published studies that scrutinize alternatives to its widely-deployed stack algorithm. In this paper, we studied an alternative Back algorithm using recency with duplicates removed.

While we began the experiment with particular expectations (framed as hypotheses), some of them proved incorrect. From our results, we now claim that people have a poor model of both stack and recency-Back. We also claim that in everyday use, people do not mentally predict what pages will appear as they click Back. Rather, we suggest that people employ a very simple 'click until recognize' strategy, where they simply click Back until they recognize the desired page. We also claim that people have no strong preference of recency or stack.

For browser designers, we advocate the replacement of stack-based Back with recency only if other design considerations warrant them. We feel that good design opportunities do exist, especially for a recency-based Back to be integrated with a recency-based history list to produce a single model of how pages can be revisited.

There is no question that the high usage rate of Back warrants further research: millions will be affected by even a small improvement in its design.

Acknowledgements. Microsoft and NSERC funded us. Kent Sullivan, Robert Graf, Linda Tauscher and Andy Cockburn contributed intellectually.

References

[1] Cockburn, A. & McKenzie, B. What do web users do? An empirical analysis of web use. *Int J Human Computer Studies* **54**, 903-922, 2001.

[2] Cockburn, A. & Jones, S. Which way now? Analysing and easing inadequacies in WWW navigation. *Int J Human-Computer Studies* **45**(1), 105-129, 1996.

[3] Cockburn, A. & Jones, S. Design issues for World Wide Web navigation visualisation tools. *Proc RIAO'97: The 5th Conference on Computer-Assisted Research of Information*. McGill University, Canada, 55-74, 1997.

[4] Greenberg, S. & Cockburn, A. Getting back to Back: Alternate behaviors for a web browser's Back button. *Proc 5th Annual Human Factors and the Web Conference*, NIST, Gaithersburg, USA, 1999.

[5] Kaasten, S. and Greenberg, S. (2001) Integrating Back, History and Bookmarks in Web Browsers. *Extended Abstracts of ACM CHI'01*, 379-380.

[6] Tauscher, L. & Greenberg, S. How people revisit web pages: Empirical findings and implications for the design of history systems. *Int J Human Computer Studies*, **47**(1), 97-138, 1997.

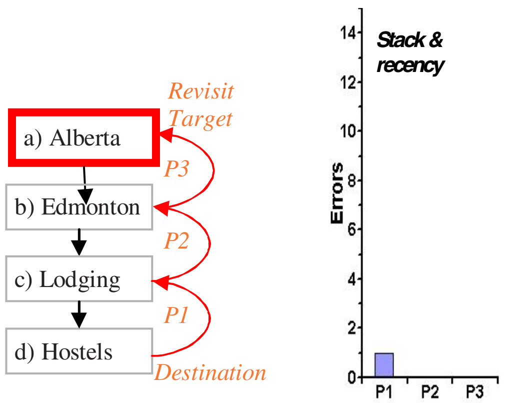


Figure 5: Short Linear

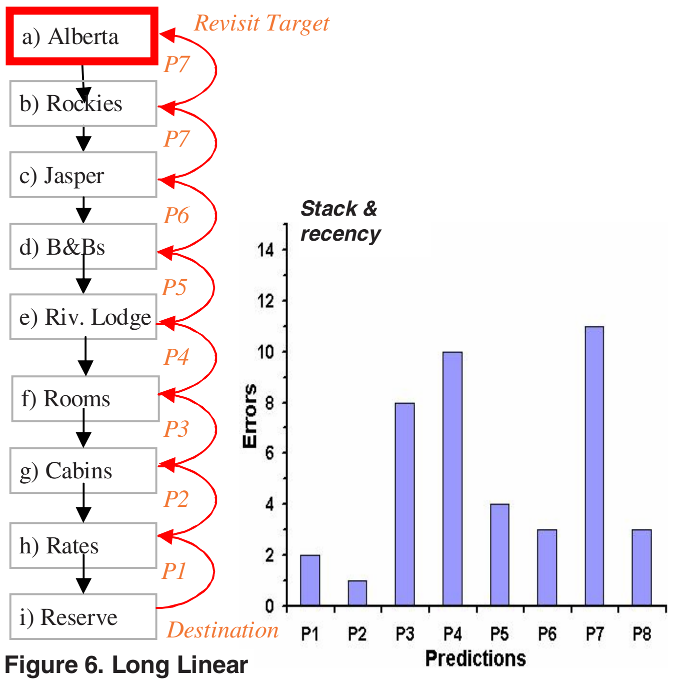


Figure 6. Long Linear

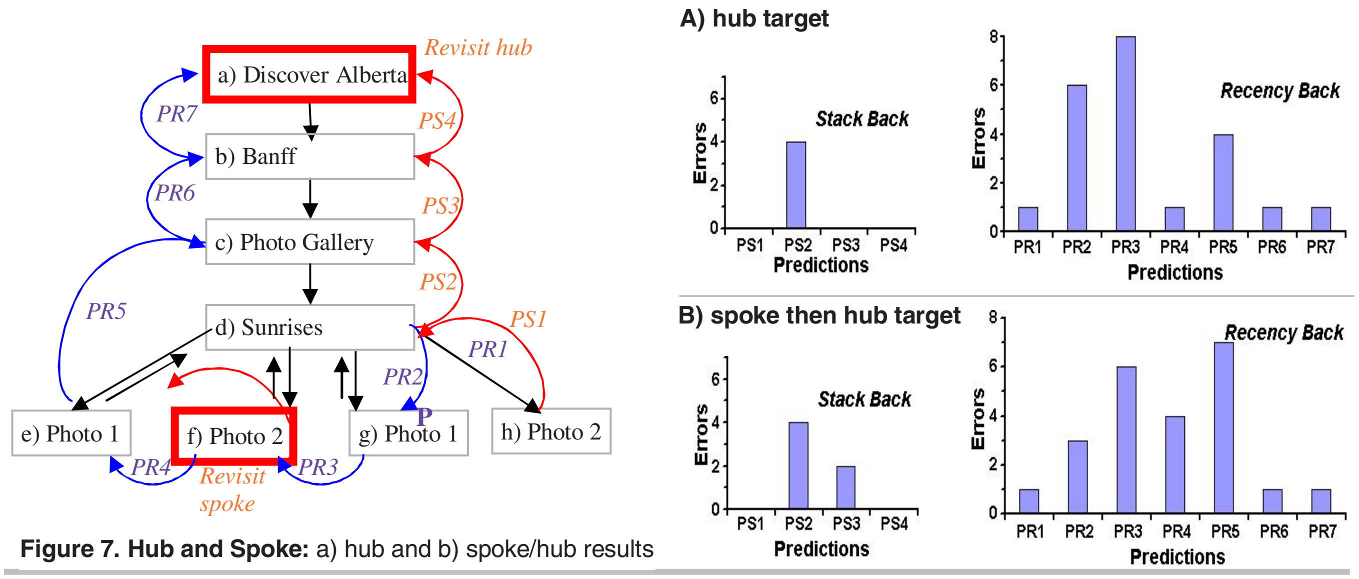


Figure 7. Hub and Spoke: a) hub and b) spoke/hub results

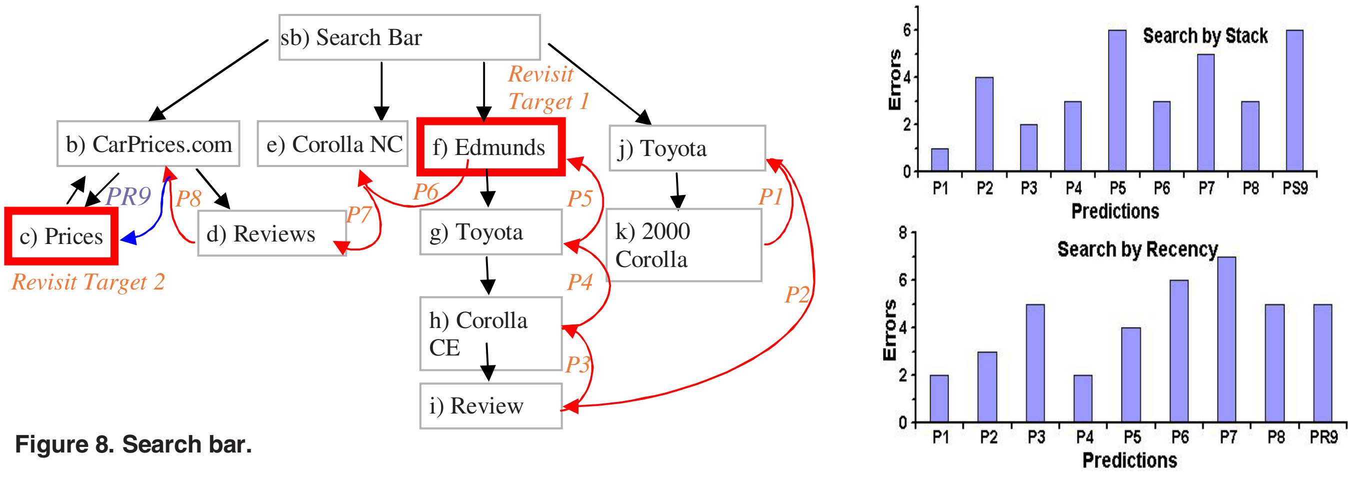


Figure 8. Search bar.

9. Appendix 2: Experiment Consent Form

UNIVERSITY OF CALGARY

Department of Computer Science Consent Form

Title of Investigation: Thresholds for Recognition of Web Pages by Various Representations

Investigators: Shaun Kaasten, Christopher Edwards, and Saul Greenberg

This consent form is only part of the process of informed consent. Please take the time to read this ask about anything you are not certain about.

Description of Research Project:

The focus of the study is to determine how web pages can be represented to make them recognizable and easily accessed in a web browser program. To do this, we are studying how well people are able to recall aspects of a web page when given its title, web address or a postage-stamp sized image of it.

First, we will give you a program that will collect a listing of the web pages your web browser has visited. This record will then be used to randomly select a sample of 30 pages.

For each of these selected pages, we will show you its title, web address or a miniature image of it. At first you will only see a small portion of it, but the portion grows until you stop it. We want you to stop the growing when you are able to recognize some aspect of it – perhaps, for example, the name of the web site it was taken from. Once you have described this, the growth continues until you are able to recognize the exact page. If you can't recognize it at it's largest size, we will ask you to take a guess.

After you completed this for each page, we will show you the actual pages in a regular web browser and ask if your initial guesses were correct. We will also ask you questions about what aspects of the web page made it easy or difficult to recognize.

The entire study requires about one hour to complete.

Our analysis will not reveal the particular web sites you have visited, although this information will be available to us in order to collect the images for the study. Our analysis may discuss the aspects of pages, for example if images are contained on them. Page type or content will not be used to analyze the data collected within the study.

Participation in this study will not put you at any risk or harm and is strictly voluntary. All information regarding your personal information and your performance is confidential and only the researchers involved with have access to it.

Informed Consent:

By signing this, you are indicating that you understand this information and agree to participate. You are free to withdraw at anytime without penalty. If you have further questions contact:

Shaun Kaasten, Department of Computer Science, University of Calgary
kaasten@cpsc.ucalgary.ca

Chris Edwards, Department of Psychology, University of Calgary
cjedward@ucalgary.ca

Saul Greenberg, Department of Computer Science, University of Calgary
Phone: (403) 220-6087, Fax: (403) 284-4707, saul@cpsc.ucalgary.ca

Participant	Date
Investigator/Witness (optional)	Date

A copy of this consent form will be given to you to keep for your records if you request it. This research has the ethical approval of the University of Calgary.