Scientist's Workbook

5th November 1999

Michael JasonSmith

Abstract

This project discusses the scientist's workbook, a system designed to assist scientists' in their work. An overview of similar systems, such as the Memex and Xanadu, is presented. The results of a series of interviews with nine scientists are discussed. In the interviews the scientists expressed difficulty managing bookmarks, user defined links into the World Wide Web. This lead to the design and implementation of a system (SWB) to assist scientist with web revisitation. A number of design goals for systems that implement web-page revisitation are given, and the implementation of SWB is detailed.

Contents

Chapter 1

Introduction

Scientists do not use computers as much as would be expected. The experiments, proofs, and musings of the scientist are often hand written in workbooks and scraps of paper. Journals, articles, and photocopied sections of books are filed, and then lost. The primary goal of this project was to examine what functions are needed of an electronic scientist's workbook, and how such a workbook would integrate with scientists' work practices.

Chapter 2 examines some of the previous systems that have been designed to assist scientists in their work. These include the Memex, proposed by Vannevar Bush in 1946, NLS, Xanadu, and the World Wide Web. All these systems were developed by scientists to assist them with their work. Despite the work done by Bush, Englebart, and others, modern computers are not ideal for scientists. Section 2.2 looks at two alternative user interfaces, the post-Wimp user interface (Section 2.2.1) and pen based input.

To better understand the problems that scientists had with their systems, a series of interviews with nine scientists from the Department of Computer Science at the University of Canterbury was held (Chapter 3). The interviews — based on a similar series of interviews conducted by Thomas W. Malone in 1982 — investigated the scientists' work practices by questioning the scientists about their work environment. These interviews identified two specific problems: organization of data that is collected as part of their research, and organization of bookmarks (user defined links to web pages).

The project focuses on the problems with bookmarks. Bookmarks are a particular method of revisitation, and have similar functionality to command-line histories and image viewing programs (Section 2.3). An analysis of a number bookmark files is carried out to determine the extent of organisation and disorganisation of the bookmark files. It was discovered that the mean folder size for a bookmark file is between 7.1 and 7.7 bookmarks per folder.

The two specific problems that scientists had with bookmarks is that they were hard to organise, and pages often changed. To overcome these problems a system called SWB was developed (Chapter 4). SWB allows the scientist to organise his or her pages with minimal effort, by providing implicit visit capture and multiple views. A number of design goals specific to SWB, but generalisable to other systems that require history revisitation, are presented. The implementation of SWB, and its architecture which is based on a caching proxy-server, is described in Section 4.2.

Finally the results are summarised in Chapter 5.

Acknowledgements

Many thanks to Dr. Bruce McKenzie for his ideas and the program that allowed the Netscape Navigator database to be examined (allowing for Figure 4.4 to be created). Thanks must also go to Dr. Andy Cockburn for keeping the project on track. Jane McKenzie is responsible for all coherent sentences in this project — the rest can be blamed on Michael JasonSmith. Thanks to the scientists who agreed to be interviewed, and to Dr. Paul Ashton and Dr. Antonija Mitrovic for their contributions (that never made it into the final report). Finally, thanks to honours class of 1999 for...being the honours class of 1999.

Chapter 2

Previous Work

> The investigator is staggered by the findings and conclusions of thousands of other workers — conclusions which he cannot find time to grasp, much less remember, as they appear.
>
> Vannevar Bush, *As We May Think*, 1945

2.1 Early Systems

The history of systems designed to assist scientists with their work is one of the oldest in Computer Science. In 1946 Vannevar Bush first proposed a system that allowed users to link pages to other pages. The idea of associative links between pages was further developed by L.C.R. Licklider and Douglas Englebart during the 1960–1980s when early hypertext systems such as NLS (oN Line System), and Xanadu were developed. While the early hypertext systems had an effect on the computer systems that scientists used, it was not until 1990 that the first system was developed that allowed scientists to link and distribute data in a way similar to that proposed by Bush 44 years earlier.

2.1.1 Memex

As the Director of the United States Office of Scientific Research and Development, Vannevar Bush oversaw the research efforts of six thousand scientists during World War Two. Because of the large number of publications produced by researchers during this period, Bush had difficulty trying to keep track of the research efforts. Bush feared that important knowledge may be lost — in the same way as Mendalian genetics "was lost to the World for a generation" [1]. To comprehend the "findings and conclusions of thousands of other workers" Bush proposed a hypothetical device called the memex; "a device in which an individual stores all his books, records, and communications" [1]. The initial draft of 'As We May Think' was written before the start of the war, but it was not published until 1946 when the quantity of data produced by the scientific community was very large [2].

Bush proposed that the memex would collate all the scientist's documents and provide links between related articles, forming an association between

documents that was more like the association in the human mind. Bush proposed that the scientist worked with a small camera attached to his or her head, and photographed documents for later addition to the desk-sized memex. When the new document had been added to the memex, the scientist would create links to the document so he or she would be able to find the document in the future. Bush also proposed a system for allowing scientists to share collections of documents with each other, annotate documents, and store previously followed information trails. Bush even suggested that document collections could be distributed with ready-made trails through the collection, saving the scientist the trouble of discovering related documents.

While never created, the memex was influential in the design of later hypertext systems. Bush's vision was extended by J.C.R. Licklider who became the first head of the Information Processing Techniques Office at the Department of Defence Advanced Research Projects Agency (DARPA). During the 1960s Licklider assisted many of the early researchers in the area of information retrieval and processing by provideing funding from DARPA. One such benificary was Douglas Englebart, who researched Augmenting Human Intellect (Section 2.1.2) [2]. Later the World Wide Web created the fullest realization of a memex-like system, with scientists creating arbitrary links between documents, and companies such as Yahoo create ready-made trails through related information.

2.1.2 oN Line System

At the end of the Second World War Douglas Englebart read As We May Think and decided to extend Bush's ideas. Englebart proposed that a computer could do "some of [the scientist's] symbol manipulating processes" so the scientist could concentrate on more abstract concepts [2]. Englebart saw "knowledge work as building structural concepts" in an attempt to help people to cope with the increasing complexity of life [3]. Englebart started creating his prototype system, called NLS (oN Line System), in 1963. NLSused an early word-processor to create documents, a graphics program to create images, and linked documents together using a hypertext system. Many features of modern computers were pioneered by NLS, such as mice, multi-windowed displays, and electronic meeting rooms. NLSwas developed into a commercial system by Honeywell but it was not a commercial success; however Mark Postel used NLS to manage the RFC(Request For Comment) standards that documented the standards that governed the early Internet.

Englebart's main concern was with making systems that allowed people to better cope with the increasing complexity of the information. Englebart wanted NLS to provide facilities that were not able to be provided by existing systems. To this end he did not necessarily want NLS to be easy to use. Englebart drew a simile between computer systems and a cycles — stating that a tricycle is easier to use but a bicycle is faster [3]. Englebart was concerned that NLS was able to link arbitrary documents, a theme that runs through the work of Bush, and Berners-Lee also.

2.1.3 Xanadu

One of the earliest attempts at a hypertext system was the Xanadu project. Xanadu was started in 1960 and was actively developed until 1992, when the funds from Autodesk were withdrawn [4]. The features of Xanadu were similar to NLS and the World Wide Web, but it included features such as bidirectional links, link consistency, and authorship control which were not present in either system. Xanadu was unsuccessful for two reasons: the code to allow people to use Xanadu was not released until 1999 [5], and had difficulty scaling to a network the size of the Internet due to the large database needed to control the bidirectional links.

2.1.4 World Wide Web

The basic World Wide Web protocols were created by Tim Berners-Lee in late 1990, initially to implement a phone-book system for CERN, the European particle physics laboratory. The main purpose of the web was to create a system for project coordination by providing a "shared information space" that was portable between different systems, with information stored on many servers [6]. The result was a hypertext system that was generic, distributed, and extensible. The early system had an integrated browser and editor which allowed the user to add notes, correct mistakes, and add links to the page while browsing. Later versions of the browsers dropped the editor.

Earlier hypertext systems, such as Xanadu, maintained a central database of links that kept links consistent. The central database was a limiting factor for Xanadu, as the cost of maintaining the database became too great when the system expanded. The World Wide Web did not guarantee that a link between documents will be consistent — this is the cause of the infamous `404: Not Found` errors — but gained the ability to arbitrarily increase in size. Another advantage that the World Wide Web had over previous hypertext systems was the ability to link arbitrary computer-data.

2.2 Alternative User Interfaces

Scientists do not use the current user interfaces as much as would be expected. In this section we will look at two alternative interfaces, post-WIMP user interfaces (Section 2.2.1) and pen input (Section 2.2.2), with the hope that the proposed interfaces can give some user-interface guidelines for the scientist's workbook.

2.2.1 Post-Wimp User Interfaces

Many user interfaces are based on the WIMP paradigm, where Windows, Icons, Mouse, and Pull-down menus are used to interact with the system. WIMP systems are easy to learn as they do not require the user to remember commands, or key sequences. Andries van Dam claims that such systems work well for three types of users: illiterate children, managers, and non-professional home users [7]. The WIMP method of interaction can limit the expressive power of the interface as the user spends too much time using the interface rather than

achieving his or her task. This is the reason the 'expert' users often prefer command line systems such as Unix shells, and mark-up systems such as LaTeX which allow them to better express concepts [8]. Command-line are often criticised as being difficult to learn but "for power users the concern is less with the learning curve than with the effort required to be highly productive" [7]. Scientists are 'power users' by Andries van Dam's definition, therefore any system that is designed to assist scientists in their work should not have 'easy to learn' as a primary goal, rather it should provide power facilities that allow the scientist to work more effectively. A similar statement was made by Douglas Englebart when he said that computers should be better to use, not necessarily easy to use [3].

2.2.2 Pen Input

A computer scientist is often situated at a desk where he or she conducts experiments, analyses results, and prepares reports. Other scientists are not as desk-bound as the computer scientist. Consider the geologist, biologist, or engineer who has to go out into the field to conduct experiments. Even the computer scientist is removed from the workstation when he or she attends a meeting, goes to a seminar, or conducts an interview. In many situations the scientist works away from his or her desk, and then paper notebooks, pads, or scrap paper is used, not a portable computer. The scientist's workbook should be portable. A proposed alternative to the desktop environment is ubiquitous computing [9]. There are three sizes of ubiquitous computers: boards which are large computers the size of a white-board, a pad similar in size to a textbook, and tabs which have similar dimensions to a pager [10]. The current workbooks used by scientists are similar in size and function to the ubiquitous pad computers. A pad-computer would be able to be picked up, moved, written on, and shared in the same way that a notebook is used. Pads do not have keyboards, as the provision of a keyboard would require the size of the pad to double. An alternative to keyboard input is pen input, which provides a "specific, precise, [and] flexible" interface [11] similar to pen and paper. A disadvantage of pen input is that it requires handwriting recognition.

Handwriting recognition is difficult as handwriting is different from person to person, an individual's handwriting changes over time, and each character is not easily distinguished from adjacent characters. Systems such as the Apple Newton attempted to recognise handwriting but failed to achieve the accuracy required to become useful. To try and overcome some of the difficulties associated with handwriting recognition Goldberg and Richardson developed Unistrokes [12]. The Unistroke alphabet was designed so that each character was quick to write, distinct in 'sloppiness space', and created with a single stroke. The Graffiti alphabet was based on a modified version of Unistrokes that used strokes similar to capital letters, but retained the restriction that most characters are formed by a single stroke. Graffiti was easier to learn, but slower to write compared to Unistrokes [14]. Graffiti and Unistrokes provided a means of accurately providing data to a computer system that has no keyboard. Devices such as the 3Com Palm Pilot — which is similar, in size, to a tab — became successful by combining a pen-based input with a small number of tools that help personal organisation. The Palm Pilot made use of a specific writing area that is used to write all characters. This allows for better

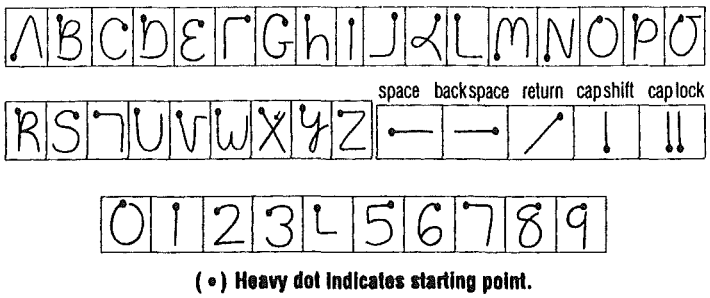


Figure 2.1: Graffiti Alphabet [13]

character recognition as each character is easier to distinguish.

Ideally the scientist's workbook should be able to be used in all environments that scientists work. The hardware to allow the workbook to recognise handwriting will be unavailable in the near future; an intermediate solution would be to use a simplified handwriting system such as Graffiti or Unistrokes as these systems have already proven themselves in real-world applications.

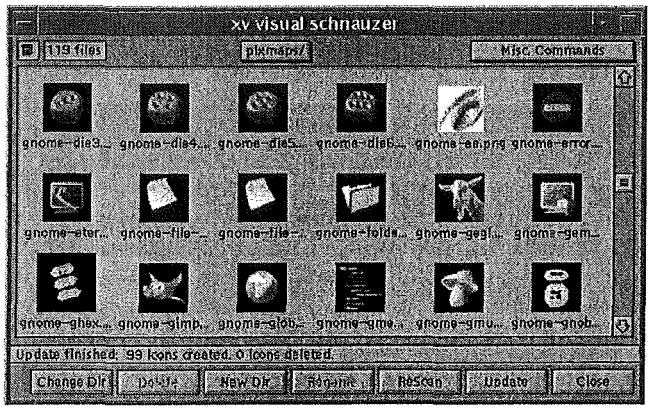
2.3 Existing Information Visualisation Systems

The scientist's workbook encompasses all work that the scientist undertakes in his or her life. The scope of the scientist's workbook is larger than is possible to cover in an honours project. At its simplest level the scientist's workbook is a tool to assist the scientist in his or her work; like any took it exists to solve specific problems. A series of interviews was carried out to determine what problems scientists experienced, with the intention to find a single area that the reminder of the project could focus on (Chapter 3). The project will now cover bookmarks, which many scientists identified as a problem area (Section 3.1.3). To better understand the problem that scientists have with bookmarks, various bookmarking and revisitation systems will be examined.

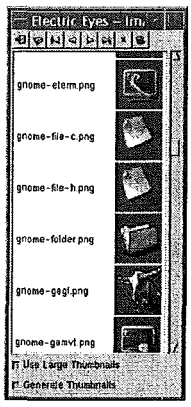
Bookmarks are a method for users of the Internet to organise links to specific URLs. They provide quick access to key sites, make it easy to return to frequently visited sites, and may provide a method of recording the history of a browsing session [15]. They share many similarities with other revisitation schemes, such as command-line histories, and image viewers. Section 2.3.1 discusses the different methods that image viewers use to view collections of pictures, Section 2.3.2 examines the methods that command-line interfaces use to repeat commonly used commands. Finally, Section 2.3.3 looks at the methods that traditional web-browsers use to revisit web-pages.

2.3.1 Image Viewers

The purpose of image viewing programs is to allow a user to scan multiple images to find one particular image. In some respects image viewers are a specialised version of a file manager, where a thumbnail of an image is used



2.2.1: XV



2.2.2: Electric Eyes

Figure 2.2: Separate Window Image Viewers

instead of a generic iconic representation of an image. When a user selects a thumbnail, a larger version of an image is shown. The primary reason that image viewers are being examined is that they provide a canonical example of GUI file-viewing tools, while also providing simple file management similar to email-programs, bookmarking systems (Section 2.3.3), and file managers such as Windows Explorer and MacOS Finder. An image viewer should allow the user to quickly scan thumbnails and select the image that he or she is seeking.

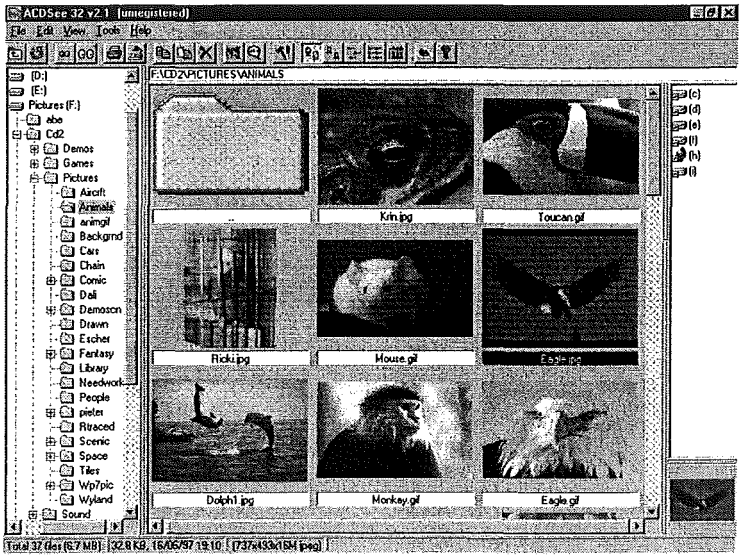
Image viewing programs use two methods to visualise collections of images. XV (Figure 2.2.1) and Electric Eyes (Figure 2.2.2) display thumbnail images in one window, while the selected image is displayed in a separate window. The split of the thumbnail and image windows allows the image window to scale to arbitrary size while allowing the thumbnail window to stay comparatively small. File management is mainly performed by direct manipulation, where a thumbnail is dragged onto a new location or where a action is selected from the tool-bar.

ACDSee (Figure 2.3.1), and its Linux Clone GTKSee (Figure 2.3.2), have an interface that is similar to many file managers. Unlike Electric Eyes and XV the thumbnails are in the same window as the larger image; this requires less window management. Quick access to different folders is achieved by selecting a folder from the folder tree. Moving images between directories is possible buy dragging and dropping selected thumbnails.

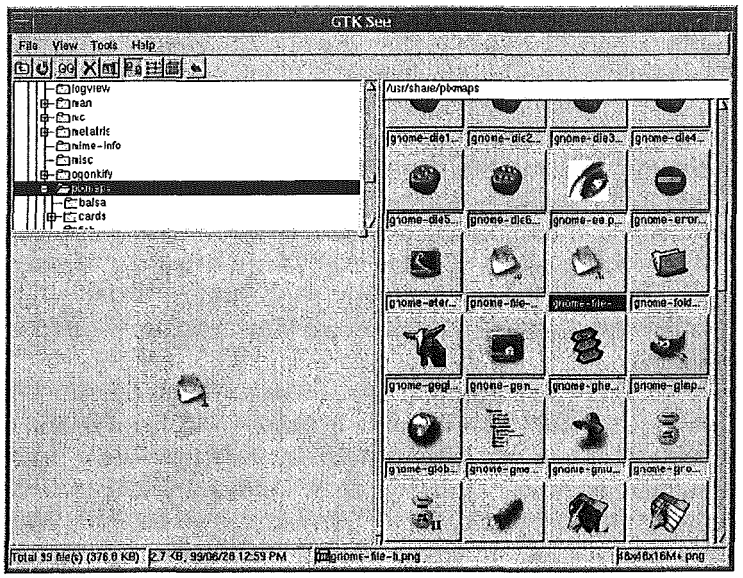
2.3.2 Command Line Interfaces

Command line interfaces are popular with power users as they allow for many actions to be performed quickly, and actions to be grouped together as scripts [8]. As the scientist's workbook is aimed at power users it is important that these interfaces are examined.

Command line shells such as Z Shell, Bash, and the Korn shell use a history



2.3.1: ACDSee



2.3.2: GTKSee

Figure 2.3: Combined image and thumbnail image viewers

```
pc62\:buffy_stuff$ cd
(reverse-i-search)`less h': less hist.tcl
```

Figure 2.4: Dynamic Search in Bash

system to allow the user to recall previous commands. Unlike web-browsers, where the visit history is a stack, each executed command is added to a list without removing the previous command. In all three shells the command history list is visualized using the `history` command. Users browse the history list by typing a keystroke, such as up-arrow, to go to the previous command. The shells have the ability to search the history list, searching for substrings in previously executed commands. Bash and Z Shell provide a dynamic search that displays the closest match to the current string (Figure 2.4).

Shells also provide a system that allows users to revisit common directories, providing functionality similar to bookmarks. A tilde (~) before a name instructs the shell to expand the abbreviated path. This is done by looking up the list of home directories and locating the user with that name. The name is then substituted with the users home directory. For example `~mpj17` is expanded to `/users/cosc/honours/mpj17`. Use of tilde-expansion is limited, as only the system administrator can add a new home-directory. An alternative to tilde-expansion is the `alias` command. Normally `alias` is used to provide quick access to frequently used commands, for example `ls -l` is often aliased to `ll`. This is similar to bookmarks when commands, such as `alias project="cd ~/project/writeup"`, are used to allow quick access to directories.

2.3.3 World Wide Web Browsers

The World Wide Web is a network of computers that allows users to remotely browse the documents on a computer, and allows documents to arbitrarily link other documents on the same network. The linking of associated documents — a feature first proposed by Vannevar Bush in 1946 — allows documents to better match the mental image that the user has of the association between documents. A difficulty that users experience with the Web is the tangled nature of the links. The paths through a series of documents on the Web can be difficult to remember, so the popular browsers provide a number of features that allow a user to revisit pages that he or she has seen in the past.

Table 2.1 compares the revisitation functions of the two most popular browsers: Netscape Navigator and Microsoft Internet Explorer. The two primary methods of revisitation are by explicitly marking a page as important (bookmarking), or by reviewing the browsing history.

While scientists interviewed only used Navigator (Section 3.3), Table 2.1 shows that both Navigator and Explorer do not differ significantly in functionality, so the problems that the interviewed scientists experienced with Navigator can be generalised to include both systems.

While Navigator and Explorer are both used in a highly visual environment, neither provide a bookmark management system that is as visual as a image viewer. Thumbnails or iconic representation of pages are not provided by either systems, and the user must use the titles given to each bookmark.

The bookmarking systems do not make use of the advantages of text in the same way as command lines. The titles are not able to be searched and the history list is not accessible by using the keyboard.

Table 2.1: Comparison of browser revisitation systems

Feature	Navigator	MSIE
Bookmark Creation		
Icon Drag	Uses page title if available	Uses page title if available
Add to Bookmarks	Uses page title if available	Uses page title if available
Right-Click on link	Uses URL	Uses anchor name
Editing Bookmarks	⟨Edit Bookmarks⟩ window	⟨Edit Favourites⟩
No Bookmark Title	Uses a shortened version of the URL with the first 18 characters, ellipsis, and then the last 18 characters	Displays the start of the URL in the ⟨favourites⟩ panel, displays the filename in the ⟨history⟩ panel
Long Bookmark Title	Uses a shortened version of the URL with the first 18 characters, ellipsis, and then the last 18 characters	Truncates the title to the width of the panel
History Visualisation		
Back button Displays stack	Yes	Yes
Typed-in location history	Yes	Yes
Other history visualisation	⟨Go⟩ menu Maintains a history database that can be accessed by the user. Database can be ordered by date, URL, title, and visit number.	⟨View⟩ → ⟨Goto⟩ menu The ⟨History⟩ panel displays a temporal ordering of sites visited, or an alphabetical listing of sites
FTP Sites		
Default Bookmark Name	`FTP Directory:` *`url`*	Site name with ⟨Add To favourites⟩ `Directory of` *`dir`* with drag and drop
History Name	`FTP Directory:` *`url`*	`Directory of` *`dir`*

Chapter 3

Work Practices of Scientists

> No man is an Island, entire of it self; every man is a piece of continent, part of the whole...
>
> John Donne, *Emergent Occasions, 'Meditation', XVII*, 1624

To establish the work practices of scientists, nine interviews were conducted with members of the Department of Computer Science at the University of Canterbury. The interviews were based on ten similar interviews that Thomas W. Malone conducted in 1982, when he attempted to discover the ways people organized their desks [16]. In Malone's interview each interviewee was asked to "give the interviewer a tour of their office" while they were asked questions about his or her environment. At the end of the interview the interviewee was asked to retrieve several documents before he or she was asked a set of predefined questions.

In the interviews for the scientist's workbook each interviewee was asked to retrieve a paper that he or she had written in the last three years. The paper was then used as the starting point for a series of questions about the scientist' s work environment, the systems that the scientist used to write the paper, and the authorship process (Appendix A).

3.1 Papers

Publication of research is an important aspect of a scientist's work, and as such, it is important that the scientist's workbook assists with the processes relating creation of papers. In general, most papers are written by multiple authors. However the systems used to create multi-authored papers are usually based on single-author systems. Issues relevant to single-authored papers will be covered in Section 3.1.1, while issues related specifically to papers with multiple authors will be covered in Section 3.1.2.

The creation of a paper usually follows these steps.

1. Scientist conducts research.
2. An initial version of the paper is written.
3. The article is submitted to a journal or conference.

4. The article is reviewed by anonymous referees who decide whether the papers is accepted into the conference or journal. If it is accepted comments and modifications are sent back to the scientist.

5. The paper is modified and then returned to the journal or conference.

6. The journal or conference publishes the paper.

7. A conference presentation, or a seminar relating to the paper is given.

The methods used by a scientist to create documents are as varied as the scientists themselves. However there are a number of systems in common, and these will be discussed in Section 3.1.1.

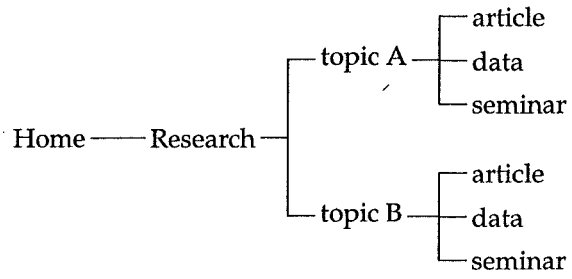
Each interviewed scientist was asked to find two papers that he or she had written, taken from the list of papers published in the 1996, 1997, and 1998 University of Canterbury Calendars. One of the papers had a single author and the other had multiple authors. The single-authored paper was chosen randomly. The multi-authored paper was chosen to have two authors that were being interviewed, so the interviewees' recollections of the authorship process could be verified. In the cases where the interviewee had not written any papers with other interviewees, the multi-authored paper was chosen to have a number of authors greater than three.

3.1.1 Single-Authored Papers

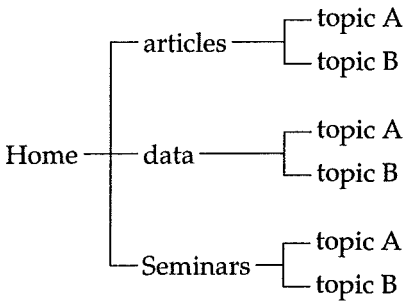
Many versions of an article are written during Step 2 of a paper's life-cycle. Often a scientist will keep draft copies to allow reference to previous versions. The paper is usually classified under research topic to allow quick retrieval. As well as draft copies of the article many scientists keep notes, printouts of graphs, and other documents related to the article.

Seven of the nine scientists used the Solaris operating system and stored electronic versions of an article in a file hierarchy. Six used topics to form a hierarchy (Figure 3.1.1), while one used directories organized by content (Figure 3.1.2). The two users of MacOS and Windows used Sherlock or Windows Finder to retrieve documents, rather than relying on a strict hierarchy to allow quick access to files. Of the scientists interviewed, all used LaTeX or Microsoft Word, but none used version control which is provided as part of Word or by an external program such as RCS (Revision Control System) or CVS (Concurrent Versions System). Instead, if different versions of an article existed, then each version would be given a unique name by the scientist.

When an article had been accepted for publication, many scientists deleted all draft copies of the paper (in the case of electronic versions) and throw out the hard-copy (if hard-copies exist at all). In many cases the version of the paper that was submitted for review (Step 3) and the final version of the paper that was accepted for publication are kept by the scientist. The majority of scientists keep only a final printed copy of the article. However some scientists only kept the journal or conference proceedings in which the article was published.



3.1.1: Topic Categorised Data



3.1.2: Content Categorised Data

Figure 3.1: Methods of Categorising Data

3.1.2 Collaboration

Most papers in the Department of Computer Science in the years 1996–1998 were written in collaboration with other scientists, with 47% of these being written by two or three authors (Figure 3.2) [17, 18, 19]. However the small sample means that a single individual could influence the numbers quite dramatically. For example, removing an author who writes many single-authored papers produced the graph shown in Figure 3.2.2.

Generally, multiple authors appeared on a paper for one of two reasons: the research was done by one author and the writeup was done by another (the research for an article is done by a student for example), or each scientist wrote a section of the paper and the sections are combined at the end. All of the scientists split a document into separate sections made use of a token passing system to control the problems of differing versions existing. In a token-passing system one author held the main copy of the paper and altered it. When the author had finished making alterations, he or she sent an electronic version to another author (passing the token) who proofread it and made further amendments, before repeating the process. Most corrections were made on printed copies of the article and then transferred into the computer. Comments for the other authors were generally placed in the document itself as either margin notes or source-code comments.

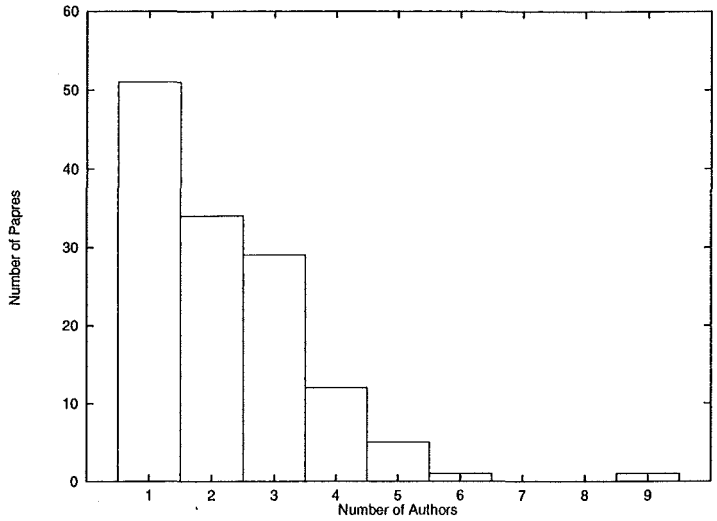
The interviewees cited few problems with existing collaboration systems. One difficulty was with multiple file-formats for the electronic version of the paper. The seven scientists in the department primarily used the LaTeX typesetting system, while two used Microsoft Word; the two systems are incompatible, causing problems when scientists collaborated.

3.1.3 Research Data

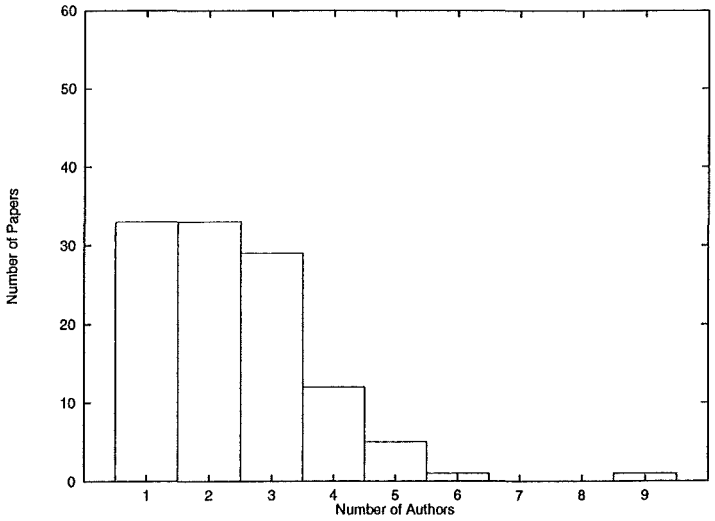
Many scientists cited problems with data that was collected as part of their research. Typical data was numeric data from timings from algorithms, network analysis, or statistical data from other sources. The numeric data was generally stored under a directory close to the article related to it (Figure 3.1.1) or in a global directory that stored all data that the scientist used (Figure 3.1.2). Data was commonly stored as ASCII text that was modified by a series of scripts before being used in tables and graphs. For example, Figure 3.4 was created from a number of bookmark files which were processed to create two summary files which were passed though an Awk script to extract the relevant fields, and finally plotted in Gnuplot. Typically file systems do not record relationships between files, the order of file creation, and the method of creation. Scientists often expressed difficulty remembering the method used to generate the files.

3.2 Books and Documents

While many scientists claimed that the Internet was becoming an increasingly important resource in their research, books remain a critical source of information. All interviewed scientists organized the bookshelves in the same way. Each had four bookcases (Figure 3.3).



3.2.1: Comparison with the single author outliers



3.2.2: Comparison without the single author outliers

Figure 3.2: Comparison of the Number of Papers with the Number of Authors, 1996–1998 [17, 18, 19]

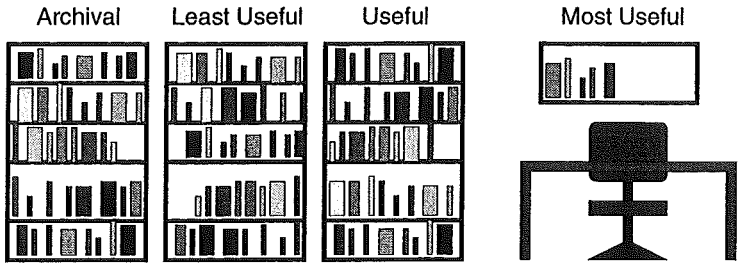


Figure 3.3: Typical organization of bookshelves

- The smallest shelf was either on, or by, the scientist's desk. It held four or five books that were referred to often, such as a dictionary, the University Calendar, the LaTeX Companion, and some course textbooks.
- The free-standing bookcase that was closest to the desk had the majority of the books that the scientist found useful in his or her work. Typically course textbooks, and seminal works in the scientist's area of research were placed in this bookcase.
- The least useful books were placed in the next bookcase. A number of these books were given to the scientist by publishers who wanted their book used as the course textbook.
- The bookcase closest to the door (and furthest from the desk) held the books that had little practical use but were kept for historical reasons, general interest, or because the scientist had a sentimental attachment to the book. Old operating systems manuals, textbooks that are no longer in use, and guides to old languages were placed in this bookshelf.

While the bookshelves themselves were classified, the location of the books in the shelves were generally governed by a move-to-front heuristic where the books that were read recently were closer to the scientist's desk. These observations are consistent with those of Card et al. [20]. Notes were a common artifact in scientists' offices. They were used to provide short messages about a research topic or to provide a reminder about an idea that the scientist had. Many scientists had difficulty with notes as they were commonly written on small pieces of paper that were misplaced easily. One scientist used a workbook to organize notes, while most had dedicated boxes or folders that contained ideas. These scientists usually had a "cleanup day" where they would read all the notes and decide what to do about them.

3.3 Network-related Data

None of the scientists interviewed stated that they had problems coping with email that they received from students, mailing lists, and collaborators. Most of the email was kept and was classified under the sender or the topic.

An area that scientists expressed problems with was bookmarks. These are a system of user-defined links to key sites. A bookmark was accessed via

a menu in the case of Netscape Navigator, or a 'panel' in Microsoft Internet Explorer. Generally bookmarks were created to allow a user to return to a frequently visited site, or a site that the user thinks will be important in the future [15]. Bookmarks allowed a user to revisit a site without remembering its URL. They can also be used to form a history mechanism that allows a user to recall the pages that he or she has been to in the past.

A significant problem that many scientists mentioned was organizing bookmarks. Netscape Navigator — which was the primary browser for all the scientists that were interviewed — displayed the data as a menu with sub-folders used to create a hierarchy similar to that of a file system. However a number of scientists had difficulty with the hierarchical menus produced by Netscape, and often lost bookmarks in sub-folders after misfiling the bookmark. Other scientists did not use sub-menus and used a large linear list that was manually searched when a link was needed. In their 1996 study, Abrams et al. discovered that 37% of the users surveyed kept the bookmarks as an unordered list. Another problem with bookmarks was caused by sites moving the page that the bookmark referenced. When a page moved, it took a great deal of effort to relocate the page, and a number of scientists stated that they were often unable to find a page if it had moved. Scientists also had difficulty determining whether a site would be useful or not. If a site proved to be useful at a later date, and it was not bookmarked, the site could be difficult to find as the history mechanism provided by Netscape Navigator was difficult to access and could not be searched. If a site was bookmarked, and was not useful, it created a superfluous entry in the bookmark menu, exacerbating the problem of locating bookmarks in the list.

In an attempt to further understand the difficulty that users experienced with bookmarks, staff of the Department of Computer Science were asked to send in their bookmarks file to be analysed for statistical properties. Seven bookmark files were sent to the author. The default Netscape Navigator bookmark file was also included in the sample. The number of bookmarks, the number of folders, the maximum folder depth, the average number of items in a folder, and the maximum number of bookmarks in a folder were examined for each file. A further 21 bookmark files were retrieved from the Internet to provide more data-points so more general results could be gained. The bookmark files were found by entering the search-strings `Personal Toolbar Folder` and `bookmarks for` into the Google search engine [21]. The two strings were selected because they occurred in all Netscape Navigator bookmark files and were uncommon in other web pages. The pages returned by Google were examined to determine whether the page continued the HTML (Hypertext Markup Language) comment `<!DOCTYPE NETSCAPE-Bookmark-file-1>` that identified the file as a Netscape Navigator bookmark file.

A comparison of the number of folders with the number of items, is shown in Figure 3.4. The best fitting linear regression for the data was $\mathit{Folders} = 3 + 0.13 \times \mathit{Items}$. This compares with the linear regression of $\mathit{Folders} = 1.14 + 0.14 \times \mathit{Items}$ reported in Abrams et al.[15]. The linear regressions equate to a mean folder size of between 7.1 and 7.7 bookmarks per folder. This plot provides a visual indication of the organization of a person's bookmarks: a highly organized bookmark collection would typically have a large number of folders in comparison to the number of bookmarks (above the linear-regression lines), while an unorganized bookmarking system would generally have a large number of items for a comparatively small number of folders (below the lines).

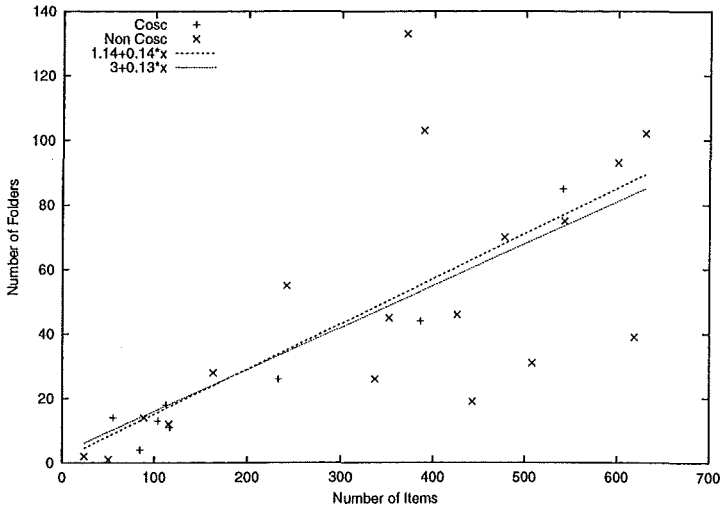


Figure 3.4: Number of Folders against Number of Items

Chapter 4

SWB

> Rome was not built in one day.
>
> John Heywood, *Proverbs, Chap xi*, Circa 1565

After conducting interviews with scientists (Chapter 3) and examining systems used by them (Chapter 2) a new system, SWB, was developed. This chapter discusses its design principles (Section 4.1) and implementation (Section 4.2).

4.1 Design Goals of SWB

The two primary goals of SWB are to implicitly capture pages, and to provide multiple views to allow users to quickly revisit pages. It is not a goal of SWB to be easy to learn, because scientists are 'power users' and as such they require facilities that allow them to work more effectively (Section 2.2), rather than to them learn quickly.

4.1.1 Implicit Visit Capture

During the interviews, scientists cited problems with the bookmarking system used by Netscape Navigator (Chapter 3). Two primary problems existed: users did not know when a page would be useful to them in the future (so they did not bookmark the page), and pages would often move or change, reducing the archival value of the bookmark.

One possible solution to this problem is to store every page that the user visits, allowing the user to recall the pages at a later date. The following design goals are a consequence of storing every page.

Implicit Storage SWB should implicitly store multiple versions of pages. When a page changes significantly the user may be unable to retrieve information that was presented on the previous version of the page. By storing multiple versions of a file the user will be able to retrieve the appropriate page.

Response SWB should cache pages without slowing down the speed of browsing to a level which users are unable to continue with their normal browsing techniques, otherwise there would be a disincentive to use the system.

Deletion It should be possible for the user to easily delete or edit web-pages after they have been captured. If a user does not want other users to know which pages he or she has has seen, the system should allow the user to remove the record of the pages, or to specify that a browsing session should not be captured by the system.

Searching The user should be able to search the stored pages for a particular HTML tag such as heading, title, or anchor as well as being able to perform a search on the entire body of a page.

Saving Searches Users should be able to save the results of a search. The time spent creating and refining a search should be acknowledged by allowing the user to save the results of the search, the parameters of the search, or both.

4.1.2 Views

SWB stores many pages and it is important that the user is able to retrieve the pages quickly. For example, if the user is looking for a page with a particular name, then the pages can be ordered by name; if the user knows that he or she visited the page at a particular time, then the pages can be ordered by time.

The following views, and properties of views, should be provided by SWB.

Temporal View Provide a temporal view, where the pages are ordered by time. Ordering views by time allows the user to quickly locate a page when the time of the visit is approximately known.

Episodes Show episodes with temporal view. An episode is a single browsing session where successive pages were viewed in quick succession [15]. Viewing episodes allows a user to quickly locate a page by being able to locate sites that were viewed in the same episode. For example, if the user can not recall the title of a page, but knew it was in the same episode as another site, then he or she can locate the known site and then locate the site in the same episode.

A combination of episodes and a temporal view provides a classification scheme and move-to-front heuristic similar to the organisation of bookshelves described in Section 3.2.

Bookmarks Identify bookmarked pages. Bookmarked pages often start browsing episodes, so identifying pages that have been explicitly bookmarked should help the user to identify episodes in temporal views [15]. Bookmarks are created because the page was considered important by the user. Because bookmarked pages are important, they should be identified in other views.

Alphabetical View Provide a name view which sorts the pages alphabetically by the title of the page. If the user knows the name, then he or she should

be able to quickly locate the page by searching through the pages which are sorted alphabetically.

Visit-Count View Provide a visit-count view that counts the number of times a host or page has been visited. A number of sites are visited more frequently than others. For example, news sites, search engines, and cartoon sites are frequently visited, and being able to distinguish these sites from other sites would be useful.

Versions Provide methods of viewing different versions of a page. While many pages do not change, and are used as an archival resource, other pages, such as news resources, change frequently. It would beneficial if the user was able to view the different versions of such pages.

4.2 Implementation of SWB

There are two components to SWB: a back-end that stores the pages for latter retrieval (SWB-WWWOFFLE, Section 4.2.1), and a front-end that retrieves the pages (SWB, Section 4.2.2).

4.2.1 Architecture of SWB-WWWOFFLE

Two alternative systems were considered to achieve the task of capturing the web-pages: the cache of Netscape Navigator could have been read, or a caching proxy-server could have been modified.

Netscape Navigator

Navigator maintains a cache of recently visited sites which is used to speed up browsing by retrieving pages from disk, rather than a remote server. Navigator's cache is made up of two components: a history database and a cache (Figure 4.1). The cache database stores information about every page that is visited, including the title, location, the time of the first and last visits, the number of times the page has been visited, and the date when the database entry will be deleted. The cached files can be accessed by reading the cache database to determine the name of the files, and then reading the raw files out of the cache. Programs such as Nscache read the cache database and allow users to retrieve files from their cache [22]. A monitoring program could access the history database and cache without the need to modify Netscape Navigator. Each time the monitoring program observed that a new page had been rendered it would have to examine the cache or history database to determine the location of the new page, retrieve the page out of Navigator's cache, and store the page in its own cache.

However a number of problems exist with this method of interacting with Netscape. While Navigator did use the standard Berkeley Database to store the history information, accessing the database was not an easy task, and SWB could not control what information was stored in the database. The monitoring program would have to either poll the browser to determine if a new page has been loaded (which would be a complex and computationally expensive task),

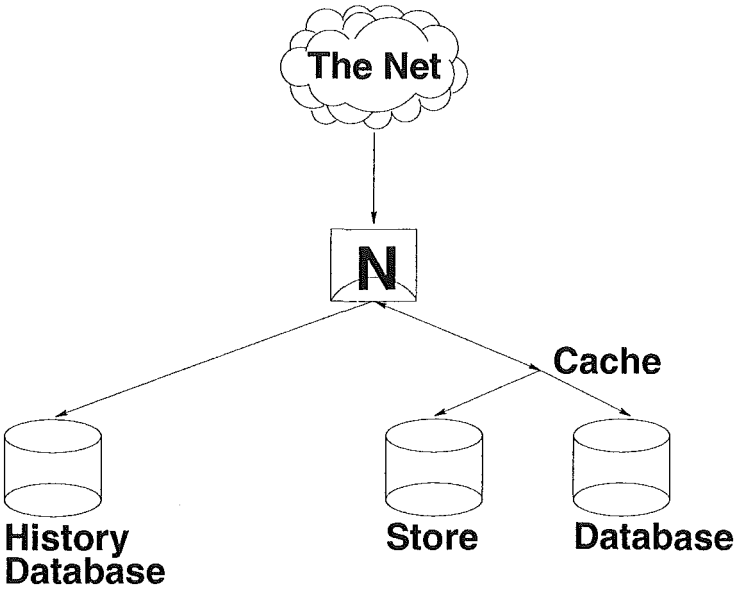


Figure 4.1: Netscape Navigator Architecture

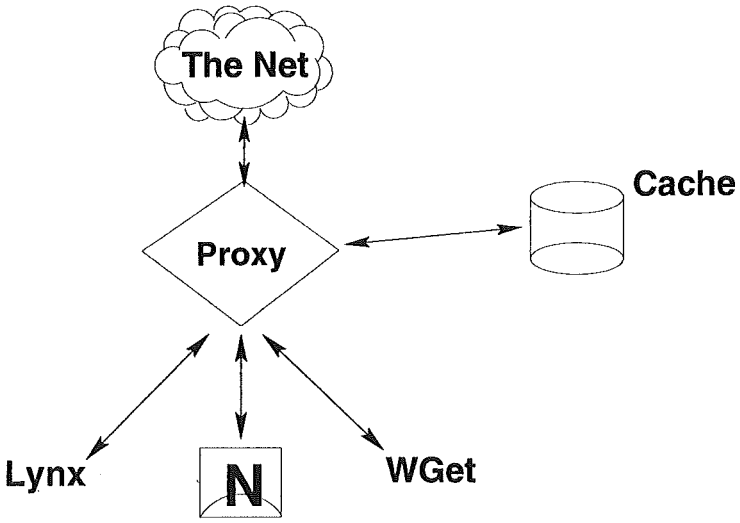
or monitor the history database and cache (which is limited by size, and is periodically flushed). Finally, any system that makes use of Navigator's history mechanism will be unable to work with other browsers.

Caching Proxy-Server

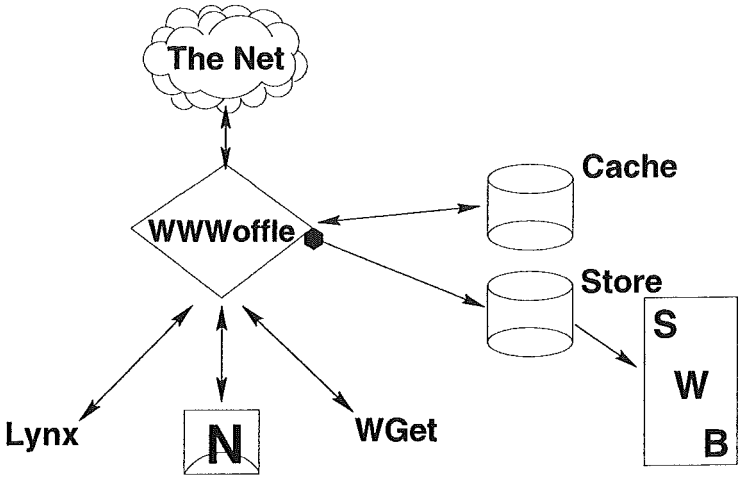
The alternative to accessing Navigator's cache is to modify an existing caching proxy-server (Figure 4.2.1). A proxy-server is a program that controls a client's access to the Internet. Commonly they are used to cache pages and files that have been down-loaded from the Internet (a caching proxy-server), block access to the Internet except for selected users (an authenticating proxy-server), or act as a filter to limit access to restricted sites. A caching proxy-server stores requested files in a similar way to that in which Navigator stores web-pages, and this store can be accessed by external programs that are running on the same machine as the proxy-server. A popular proxy server is Squid, which is used by many ISPs (Internet Service Provider) to reduce the bandwidth consumed by their users.

An advantage of running a proxy server is that it allows all client software that supports proxy servers (including Netscape Navigator, Lynx, and WGet) to become part of the SWB system. Many proxy-servers are distributed with Open Source licences, that allows the source-code to be freely modified, unlike Navigator. This allows a specialised database to be created, using all the data that is available to the proxy-server. The proxy-server chosen to be the basis of SWB is WWWOFFLE [23]. Unlike Squid, WWWOFFLE is designed to be a proxy-server for a single user; as such it is simpler than Squid but performs the majority of its multi-user counterpart's tasks.

The basic architecture of SWB is shown in Figure 4.2.2. A modified version



4.2.1: Proxy-Server Architecture



4.2.2: SWB-WWWOFFLE Architecture

Figure 4.2: Architectures

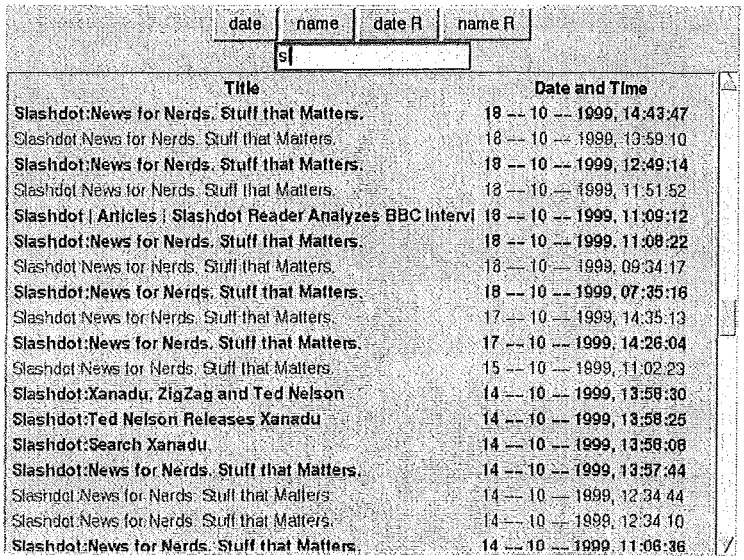


Figure 4.3: History View

of WWWOFFLE (SWB-WWWOFFLE) was the proxy-server for all outgoing HTTP (Hypertext Transfer Protocol) requests. Each time a client, such as Navigator, requests a page SWB-WWWOFFLE checks if the page is already cached. If the page is cached, the remote server is queried to determine whether the cached page is the current version. If it is current SWB-WWWOFFLE retrieved the page from the cache and sends the page to the client. Otherwise the page is retrieved from the remote server, cached, saved in the SWB store, and finally sent to the client.

The basic caching functionality of SWB-WWWOFFLE was not changed, to limit the amount of coding necessary. Instead, code was included to write each cached file to a separate store, with the file name of the cached file made up of a combination of the URL (Uniform Resource Locator) and a time stamp. This creates a unique filename that would not be overwritten by subsequent files, unlike the cache which only stores the most recent version of the file. An external program (SWB) accessed the store and communicated with a browser, such as Navigator or Lynx, to render the stored web-page. The interface of SWB is discussed in Section 4.2.2.

4.2.2 SWB: The Program

SWB provides an interface that attempts to implement the design guidelines in Section 4.1.

When the user starts SWB he or she is presented with a list of the pages that have been visited (Figure 4.3). By default the pages were sorted by date, with the most recently viewed page at the top of the list. Double clicking on an entry renders the page in a browser.

Each entry is coloured either dark olive-green or blue (shown as bold or normal text in Figure 4.3); adjacent entries that had the same colour were mem-

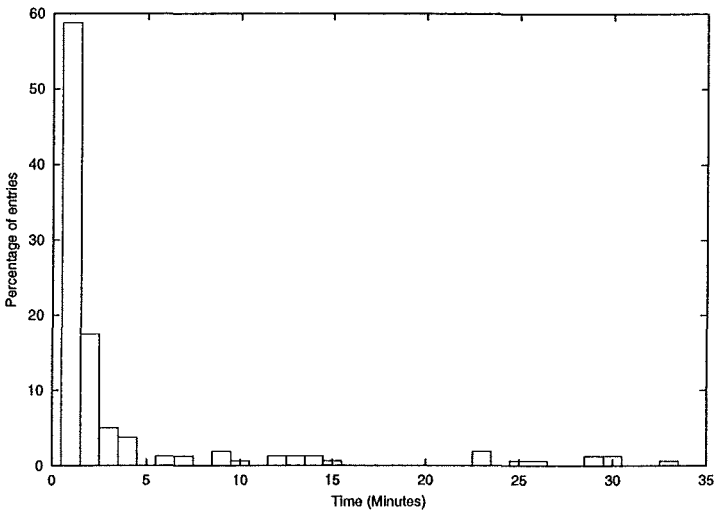


Figure 4.4: Time differences between consecutive history database entries

bers of the same episode. For example, the entries from 13:57 to 13:58 on the 14th of October are part of the same episode. Episodes are determined by calculating the differences between the viewing times of adjacent entries in the list. When these are above a threshold, SWB changes the entry colour to signify the start of a new episode. The threshold was determined by examining the author's Netscape Navigator history database and examining the time differences (Figure 4.4). The majority of time differences were one minute or less, with 85% of time differences less than six minutes. Six minutes was therefore selected as the threshold.

To search the titles of pages, the user types in the text entry-box and entries that do not match the search string are dynamically removed from the list. Often very short strings are required, for example the string `sl` matches all pages from the news site Slashdot (Figure 4.3).

4.3 Further Work

The architecture of SWB-WWWOFFLE is a prototype. As a consequence, the modifications made to WWWOFFLE were minor, and the overall architecture would benefit from improvements in many areas.

- A complete web-page usually consists of images and HTML (Hypertext Mark-up Language) text. While SWB-WWWOFFLE stores all down-loaded data, SWB does not allow embedded data (such as pictures, sounds, and style sheets) to be retrieved. The initial SWB system only retrieved HTML for two reasons. The author did not download the images from a web-page by default, so SWB not retrieving images was not noticed. Secondly, problems from determining the filename to the original URL have not been resolved. The primary difficulty with filename resolution is the HTML tags for the embedded data, which refer to the original filename

on the remote server, rather than the modified filename on the local disk. While the tags could be modified, each tag would have to be modified after the embedded data had been down-loaded because the filename is made up of the download time and URL.

- Only particular files, such as text and PDF files, contain data that can be searched, and it would be useful if SWB-WWWOFFLE recorded the type of each file in a database (this information is passed to all HTTP clients by the servers). SWB uses the Unix `file` command to determine whether a file contained data that could be searched or not.

- Communication between SWB-WWWOFFLE and SWB could be enhanced. Currently, SWB has to be restarted so it updates its internal list of pages. A superior solution would be if SWB-WWWOFFLE notified SWB when a new page had been down-loaded.

- User's sharing of cache files could be implemented. There are many ways that users could share cached files: they could browse other users' caches, swap individual cache entries, and share a single store (rather than having a separate store and occasionally browsing other users' stores). This would possibly require another proxy-server to be used, as WWWOFFLE is designed to be used by a single user. Issues relating to privacy would have to be examined before cache sharing could be implemented.

- Extending the SWB system to all files (not just web-pages). This could be achieved by either creating a new file-system that stores multiple versions of files, or by layering a virtual file-system on top of an existing file-system and using a version-control system such as RCS or CVS to implement transparent version control.

SWB does not incorporate all the guidelines proposed in Section 4.1. Deletion of entries is not possible, but it is possible for the user to specify that a browsing session is not captured by setting the browser not to use SWB-WWWOFFLE as the proxy server. A temporal view is provided, with episodes shown, and an alphabetical view is also provided. However the visit-count view is not yet implemented, and bookmarked pages are not yet distinguished. Only simple searching has been implemented, and users are not yet able to save searches. Table 4.1 summarises the design goals.

One of the primary failings of SWB is that its interface is not visually appealing. A prototype interface is shown in Figure 4.5, where the author's Netscape Navigator bookmark file is displayed in a similar manner to a directory tree. The different views are represented by the end group of icons in the tool-bar. When the user clicks on a view, a different classification scheme would be shown in the left-hand pane. The folder-file view common in file managers and image-viewing programs (Section 2.3.1), splits the classification (the folders) from the results of the classification (the files). This system could be extended to other classifications: in the case of SWB shown in Figure 4.4 the different pages orders would appear in the left-hand pane instead of the folder tree.

Goal	Goal Met
Implicit Storage	Yes
Response	Yes
Deletion	Partially
Searching	Partially
Saving Searches	No

(a) Implicit Visit Capture Goals Met

Goal	Goal Met
Temporal View	Yes
Episodes	Yes
Bookmarks	No
Alphabetical View	Yes
Visit-Count View	No
Versions	No

(b) View Goals Met

Table 4.1: Summary of design goals met

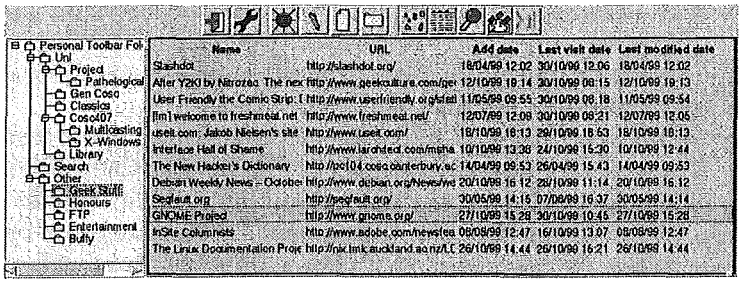


Figure 4.5: Bookmark view

Chapter 5

Summary

Beginning with the Memex in 1946 there have been a number of systems developed to help scientists with their work. Systems such as NLS and Xanadu were developed by scientists to help them to read and create documents. These systems formed a hypertext system that linked all the scientist's documents together. Hypertext became popular with the advent of the World Wide Web, that was developed by Tim Berners-Lee in late 1990, and many scientific publications have been made available on the Web.

It was argued that modern user-interfaces are based on the WIMP paradigm which does not suit scientists, as scientists are power users and require systems that allow them to be highly productive, rather than systems that are easy to learn.

A number of image visualisation and revisitation schemes were also examined.

A series of interviews with scientists took place to determine specific problems that scientists had with their work, and concentrated on the systems that scientists used to write papers. The interviewees had little difficulty with the majority of the systems that they used. However, two areas of difficulty mentioned by many scientists: organizeing research data, and managing bookmarks to web pages. Scientists had difficulty with the research data as the files often did not record how the data was generated. Bookmarks were used by many scientists, but they expressed difficulty in organising the bookmarks.

The problems with bookmarks lead to the development of SWB, a system designed to assist with the revisitation of web pages. A series of design goals for the SWB system was presented. Although these goals were created specifically for SWB, they can be extended to other systems that rely on history mechanisms, such as history lists with command-line based systems. The architecture for SWB was also covered, and the rationale for having the architecture was also given.

Appendix A

Preliminary Question Session

A.1 Opening Soliloquy

Before we begin I would like to point out that this interview can be stopped at any time, with no questions asked. You, and your work are not being assessed. The interview is confidential, and all results will be made anonymous.

The honours project I am working on is a system called the Scientist's Workbook. The idea is to develop a computerised system to replace the mixture of systems currently used. It is hoped to discover what facilities provided by such a system will be most useful to scientists.

To this end I am conducting a series of informal interviews to discover the current work-practices of scientists. Specifically I'm looking at what data scientists use, how they write papers (and similar technical documents), and what computer systems they currently use. It is hoped that from these interviews an idea of the ideal device can be gained.

The interview will last approximately half an hour.

A.2 Single Authored Article

- Ask the interviewee to find an article.
- Ask the interviewee how the article is stored.
- Ask the interviewee whether the drafts are kept.
  - If no why not?
  - If so why and how are the drafts stored?
- Ask about the information resources used to create the article, how they are kept and referenced:
  - Books
  - Articles
  - Journals
- Ask about any online resources that they may use.

- Ask about experiments, data from experiments, and the storage of the data.

A.3 Collaborative Article

- Ask how they split the article up.
- Ask whether they have problems with differing versions of the same document.
- Ask if they had trouble getting access to the other author's references and resources.
  - transmitting and sharing
  - multiple copies
  - annotations
- Ask how they communicate. What do they use, what don't they use (if not, why not).
  - phone
  - letter
  - email
  - fax

A.4 Seminar and Talk

- Ask them about the last seminar (what was it about).
- Are the resources for the seminar organized any differently than for the articles?
- How do they organize the resources for presenting the seminar (overheads, slides, PowerPoint...).
- Is there a difference between a seminar and a presentation at a conference?

A.5 Web and E-mail

- How are online resources, such as bookmarks, organized?
- Do you occasionally come across items on the Net that interest you but are not immediately useful? How is such information kept for later reference?
- Do you find bookmarks useful?
- Do you like your organization of your bookmarks?

- How is your email organized?
- Do you think that the email, bookmarks, and the directory structure bear a similarity?
- (*Possibly skip?*) Do you think that you have control over email? That is, do you have trouble classifying email?
- Do you think that your organization of your email is useful?
- Do you like your organization of email?

A.6 Other Items and Artifacts

- How are the related data stores (talk or seminar, article, data, and online resources) organized?
- How do you organize references?
- How do you deal with data that does not fit into the existing systems: that is, things like ideas and miscellaneous files (things that you come across on the web):

Bibliography

[1] Vannevar Bush. As we may think. *Atlantic Monthly*, 176(1):101–108, July 1945.

[2] Larry Press. Before the Altair: The history of personal computing. *Communications of the ACM*, 36(9):27–34, September 1993.

[3] Douglas C Engelbart. Towards augmenting human intellect and boosting our collective I.Q. *Communications of the ACM*, 38(8):30–32, August 1995.

[4] Xanadu Australia. Xanadu faq. World Wide Web, July 1997. http://www.xanadu.com.au/general/faq.html.

[5] Jon Udel. Xanadu report. World Wide Web, August 1999.

[6] Tim Berners-Lee. WWW — past, present, and future. *IEEE Computer*, 29(10):69+, October 1996.

[7] Andries van Dam. Post-wimp user interfaces. *Communications of the ACM*, 40(2):63–67, February 1997.

[8] Don Gentner and Jakob Nielson. The anti-mac interface. *Communications of the ACM*, 39(8):70–82, August 1996.

[9] Mark Weiser. The computer for the 21st century. *Scientific American*, 265(3):66–75, September 1991.

[10] Mark Weiser. Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36(7):75–85, July 1993.

[11] André Meyer. Pen computing: A technology overview and a vision. *ACM SIGCHI Builletin*, 23(3):46–90, July 1995.

[12] D Goldberg and C Richardson. Touch-typing with a stylus. pages 80–87. Addison-Wesley, 1993.

[13] 3Com Corperation. Graffitti reference card, 1998.

[14] Alsob Stewart. Inovative graffiti might actually make PDAs useable. *Infoworld*, 16(39), 1994.

[15] David Abrams, Ron Baecker, and Mark Chignell. Information archiving with bookmarks: Personal webspace construction and organization. In *CHI '98*, 1998.

[16] Thomas W. Malone. How do people organise their desks? Implications for the design of office information systems. *ACM Transactions on Office Information Systems*, 1(1):99–112, 1983.

[17] University of Canterbury. *Canterbury University Calendar*. Canterbury University Press, Christchurch, 1999.

[18] University of Canterbury. *Canterbury University Calendar*. Canterbury University Press, Christchurch, 1998.

[19] University of Canterbury. *Canterbury University Calendar*. Canterbury University Press, Christchurch, 1997.

[20] SK Card, JD Mackinlay, and B Shneiderman, editors. *Readings in Information Visualization: Using Vision to Think*. Morgan-Kaufmann, 1999.

[21] Google Incorporated. Google. World Wide Web, 1999. http://www.google.com.

[22] Stefan Ondrejicka. Nscache. Computer Program, October 1999. Available from http://www.idata.sk/~ondrej/nscache/.

[23] Andrew M. Bishop. Wwwoffle version 2.4. Computer Program, December 1998. Avaliable from http://www.gedanken.demon.co.uk/wwwoffle.