CZWEB: AN AID TO NAVIGATE THE WORLD WIDE WEB

by

Pu Tan

B.Sc., National University of Defence Technology, Hunan, P.R. China, 1985

M.Sc., Shanghai Jiaotong University, Shanghai, P.R. China, 1991

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF APPLIED SCIENCE
in the School of Engineering Science

SIMON FRASER UNIVERSITY

June 1997

APPROVAL

Name: Pu Tan

Degree: Master of Applied Science

Title of Thesis: CZWeb: An Aid to Navigate the World Wide Web

Examining Committee: Dr. Shawn Stapleton, Professor
Chair

Dr. John C. Dill, Professor
Senior Supervisor

Dr. Tomas W. Calvert, Professor
Supervisor

Dr. F. David Fracchia, Assistant Professor
School of Computing Science
External Examiner

Date Approved:

PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/Project/Extended Essay

"CZWeb: An Aid to Navigate the World Wide Web"

Author:

_____
(signature)

_____
(name)

_____
(date)

Abstract

The World Wide Web is currently the largest and fastest changing hypertext information system in the world. Modern Web browsers allow users to access this wide range of services and information easily. But the two fundamental navigational problems with hypertext still exist, namely, disorientation and cognitive overhead. Disorientation is the feel of lost in hyperspace. Cognitive overhead is the additional effort needed to use hypertext system. The goal of this thesis is to develop methods to help users deal with these problems in the Web. Such methods illustrate and simplify the underlying structure of the World Wide Web and provide sophisticated support to users for understanding, navigating through and manipulating the complex information space.

This thesis proposes CZWeb, an aid to Netscape Navigator users on the Macintosh platform. It records visited web pages and displays them in a dynamically updated map while users navigate through the World Wide Web. A new URL-based structuring approach is proposed to organize the collected information. It represents the Web structure with a network and a hierarchy based on the URL of web documents and the embedded hyperlink relationships among them. The structure is displayed on screen with an extended Continuous Zoom algorithm, which is a detail in context approach. Some extensions are made to the original Continuous Zoom algorithm, which make it possible to explicitly control the size of individual nodes. Examples include zooming a node to a specified size and maintaining a constant size while other nodes are zoomed. Those extensions not only satisfy CZWeb's special requirements (opening and closing cluster nodes, maintaining folder icons' constant size), but are also useful for other situations.

CZWeb also provides some facilities for users to recognize and access any visited Web pages easily and allows users to tailor the displayed structure to reflect their own mental models of the information space.

Usability tests suggest that the design model of CZWeb is consistent with users' mental models of the World Wide Web and that CZWeb does help users.

Dedication

For my family – Karen and David

Acknowledgments

I am deeply grateful to my supervisor, Dr. John Dill, for his invaluable advice and guidance, especially when time is very precious to him. Without his help, this thesis would not be possible.

I would like to extend my thanks to my supervisory committee member, Dr. Tom Calvert, and my external examiner Dr. F. David Fracchia, for their insightful advice and suggestions.

Several other people have played special roles and contributed to the design and evaluation of CZWeb. They are Dr. Gérald Collaud, Dr. Chris Jones, Dr. Brian Fisher, and my classmates in the graduate HCI course, CMPT882(96-1).

I would like also to express my gratitude to all other faculty members of the School of Engineering Science and the School of Computing Science, for all the courses I have taken from them. In addition, I would like to thank fellow graduate students for any help they have given me.

Last but not least, I want to thank the School of Engineering Science at Simon Fraser University for providing me with financial support throughout my study. My sincere thanks are given to Ms. Brigitte Rabold for her great assistance.

Contents

Approval ii

Abstract iii

Dedication v

Acknowledgments vi

1 Introduction 1

1.1 Hypertext 1

1.2 The World Wide Web 3

1.3 Navigation Problems 4

1.3.1 Disorientation 4

1.3.2 Cognitive Overhead 5

1.4 CZWeb Approach to Aid Navigating the Web 6

2 Literature Studies 8

2.1 Designs for Navigation 8

2.1.1 Guided tours 9

2.1.2 Footprints 9

2.1.3 Backtrack 9

2.1.4 History lists 10

2.1.5 Bookmarks 10

2.1.6 Sneak preview 11

2.1.7 Search engines 11
2.2 Overview Diagrams 12
2.2.1 Global Overview Diagrams 12
2.2.2 Overview Diagrams to Aid Web Browsers 18
2.2.3 Overview Diagrams in New Web Browsers 28
2.3 Detail-in-Context 30

3 CZWeb Design and Features **34**
3.1 Design Goal and Challenge 34
3.2 What Information is the Most Important? 35
3.3 Transforming the Web Structure 37
3.3.1 Mental Models of the Web 37
3.3.2 Managing Historical Information 38
3.3.3 Link-based Structuring 39
3.3.4 Our Approach – URL-based Structuring 40
3.3.5 The URL Structure 41
3.4 Visual Representation of the URL-Tree Structure 43
3.5 Dynamic Updating of the Overview Diagram 46
3.6 Modify the Default Structure 49
3.7 Mapping Attributes of Nodes 49
3.8 Other User Interface Considerations 50

4 CZWeb - Internal Operation and Algorithms **52**
4.1 Communication with Netscape Navigator 52
4.2 Building a URL-tree 53
4.3 Displaying the URL-Tree 58
4.3.1 Background 58
4.3.2 Review of the Original CZ Algorithm 59
4.3.3 A Simple Case: Exact Size Zoom 60
4.3.4 Case 2: Single Size Constant 62
4.3.5 Case 3: Exact Size Zoom With Size Constant 63

4.3.6 Complex Cases 64
4.3.7 Some Implementation Considerations 65
4.4 A Simple Spring Layout Algorithm 66

5 CZWeb Evaluation **70**
5.1 First Evaluation 70
5.2 Second Evaluation 72
5.3 Third Evaluation 77

6 Future Work and Conclusion **80**
6.1 Discussion 80
6.1.1 Link-based Structuring vs. URL-based Structuring 80
6.1.2 Hyperlinks vs. Temporal Path 81
6.1.3 Distinguish User's Actions 82
6.1.4 Using Web Browsers 83
6.2 Future Work 83
6.2.1 Algorithm improvement 83
6.2.2 New Features 84
6.2.3 Other work 85
6.3 Conclusion 86

Bibliography **88**

List of Figures

2.1 Drawing Based on NoteCards [Hala87] 13
2.2 Drawing Based on gIBIS [Conk88] 13
2.3 Drawing Based on Intermedia Web View [Utti89] 15
2.4 Drawing Based on HyperTEXT'87 Trip Report [Niel90] 15
2.5 Drawing Based on Thoth-II [Coll87] 17
2.6 Graphical view generated by Hy+. Printed with permission. 26

3.1 Diagram Used in Interview 37
3.2 A Typical Overview of Cluster-Page Structure 45
3.3 A Typical Overview of URL-tree Structure 46

4.1 The Complete URL Tree 54
4.2 Comparison of the Internal URL-Tree and the External URL-tree . . 55
4.3 The Original Continuous Zoom Algorithm 59
4.4 Zoom node 1 to the exact size and node 2 is resizable 60
4.5 Single Size Constant 62
4.6 Complex Case 64
4.7 Spring Layout Algorithm 68

5.1 The Structure of Test Web Site 73
5.2 Test Site Structure Created with Standard Cluster-Page Structure Version 74
5.3 Test Site Structure Created with URL-Tree Structure Version 75

Chapter 1

Introduction

Hypertext systems are complex information management systems. These systems allow people to create, annotate, link together, and share information from a variety of media such as text, graphics, audio, video, animation, and programs. Hypertext systems provide a non-sequential and entirely new method of accessing information unlike traditional information systems which are primarily sequential in nature. This chapter is an introduction to hypertext, the World Wide Web (a popular hypertext system), and the two fundamental problems of using hypertext.

1.1 Hypertext

The original idea of hypertext was first put forth by Bush [Bush45] in July 1945. He described a device called *memex*, " a future device for individual use, which is a sort of mechanized private file and library." In which an individual "store his books, records and communications and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory." He described the essential feature of memex as its ability to tie two items together.

Nelson coined the word *hypertext* in 1965 [Nels65]. He defined it as "a body of written or pictorial material interconnected in a complex way that it could not be conveniently represented on paper. It may contain summaries or maps of its contents

and their interrelations; it may contain annotations, additions and footnotes from scholars who have examined it."

In his Keynote address at Hypertext '87, van Dam [VanD88] reviewed the very early stage of hypertext development, including his own group's efforts [Carw69]. He also summarized nine areas which are important to hypertext design.

Smith and Weiss [Smit88] defined hypertext as "an approach to information management in which data is stored in a network of nodes connected by links. Nodes can contain text, graphics, audio, video as well as source code or other forms of data."

The essential feature of hypertext, as defined in recent years, is the concept of machine-supported links (both within and between documents) [Bala96]. It is this linking capability which allows a nonlinear organization of text. Hypertext is a hybrid that spans across traditional boundaries. It is a database method providing a novel way of directly accessing and managing data. It is also a representation scheme, a kind of semantic network, which mixes informal textual material with more formal and mechanized processes. It is an interface modality that features link icons or markers that can be arbitrarily embedded with the contents and can be used for navigational purposes [Conk87]. In short, a hypertext system is a database system, which provides a unique method of accessing information.

Nodes and links are the fundamental units of hypertext. A node usually represents a single concept or idea. It can contain text, graphics, animation, audio, video, images or programs. It can be typed (such as detail, proposition, collection, summary, observation, issue) thereby carrying semantic information [Rao 90]. Nodes are connected to other nodes by links. The node from which a link originates is called the reference and the node at which a link ends is called the referent. They are also referred to as anchors. The contents of a node are displayed by activating the link that references it.

Links connect related concepts or nodes. They can be bi-directional thus facilitating backward traversals. Links can also be typed (such as specification link, elaboration link, membership link, opposition link and others) specifying the nature of relationship [Rao 90]. Links can be either referential (for cross-referencing purposes) or hierarchical (showing parent-child relationships).

Niguma designed and implemented a concept mapping tool in the World Wide Web. He also conducted a small study to evaluate the effectiveness of this tool.[Nigu97] In his thesis, he claimed "The purpose of the concept map is to identify key concepts and the relationships between these concepts in an instructional setting under various levels of abstraction". He used polygons to represent students' concepts and ideas and labeled lines connected between polygons to represent relationships between concepts and ideas. He refered to the polygons as "nodes" and the lines as "links".

There are many hypertext systems. The World Wide Web is currently the most popular one among them.

1.2 The World Wide Web

The World Wide Web (Web, WWW or W3) is a hypertext-based information retrieval mechanism providing information access across heterogeneous platforms mainly connected over the Internet [Bern94]. It is based on the philosophy that information should be freely available to anyone. It uses a client-server architecture where the information resides on servers and viewers run from clients. Its architecture allows many existing hypertext systems and information bases to be incorporated as part of the web by gateway servers. The Web servers provide their data to the clients through a standard communication protocol called HyperText Transfer Protocol (HTTP). But hyperlinks can link to other servers that use different protocols (Gopher, FTP, WAIS, etc.), too.

On the Web, the word *document* refers to a piece of information that might be delivered by a web server. Documents are located by their Uniform Resource Locators (URLs). A URL is a unique address for a document on the Web. The primary format for documents is called HyperText Markup Language (HTML). HTML documents contain formatted text and images, as well as hyperlinks to other documents, which may be anything that is accessible on the Internet: sound, digital video clips, connections to a library catalog, HTML documents, or CGI programs. Users access information in the Web through Web browsers.

1.3 Navigation Problems

Hypertext systems are used in many applications because of their flexible structure and the great browsing freedom they give to users. However, this same flexibility and freedom also causes some problems.

In his widely cited survey [Conk87], Conklin pointed out two problems "that may in fact ultimately limit the usefulness of hypertext". They are *disorientation* and *cognitive overhead.*

1.3.1 Disorientation

Disorientation is "the tendency to lose one's sense of location and direction in a nonlinear document" [Conk87]. Some other researchers refer to this problem as being "*lost in hyperspace.*" It occurs when readers do not know where they are, how they got there or where they should go next.

In a book, it is possible to flip pages randomly and read material in any order you like. It is not easy to get lost, because while you hold the book you can see and feel the whole book and also approximately where you are. Furthermore, the pages are numbered, telling you exactly where you are. You can also find topics you are interested in from the table of contents or index.

However, in a complex hypertext system with thousands of nodes and links, it is more than likely that the reader will get lost. Reasons for this include:

- Large size of the information space
- Complicated graph that links different documents
- User's lack of knowledge of the overall structure of the information space. In many hypertext documents, users can only see one node at a time.

The influence of the structure of hypertext documents on the orientation problem is best illustrated by comparing browsing through a hypertext with exploring a city as a tourist. People have much less trouble finding their way in a city with a grid pattern and using numbers to name streets and avenues than in a city with unorganized streets.

Of course, exploring a city and browsing a hypertext document are not exactly the same. Getting lost while browsing through hyperspace is more probable because of the lack of continuity. As Konstantin [Kons96] states: "When we walk through a town, we look around and build a mental map of the place. Since we see everything that is in between our origin and our destination, we will experience a sense of continuity. Clicking on an anchor or typing a URL and watching the byte counter as the destination document loads onto our screen just isn't the same".

1.3.2 Cognitive Overhead

Conklin [Conk87] characterized *cognitive overhead* as "the additional effort and concentration necessary to maintain several tasks or trails at one time". The reason for cognitive overhead lies in the limited capacity of human information processing [Kahn73].

Every effort in addition to reading reduces the mental resources available for comprehension. With respect to hyperdocuments, such efforts primarily concern *orientation, navigation and user-interface adjustment* [Thur95].

For orientation, readers need knowledge about the overall document structure and must keep track of their moves through that structure. Even for smaller hypertext systems this can result in a considerable memory load if no external orientation cues are given.

Empirical studies summarized in [Dill93] revealed a correlation between comprehension and memory for location. One interpretation of this result is that memory for content and memory for spatial information are different aspects of the same mental representation, i.e., the reader's mental model. Hence, all factors that facilitate the construction of such a model by reducing mental effort or increase a model's quality by improving its completeness and consistency can be expected to affect both comprehension and orientation.

A node may provide several links to other nodes. When the number of choices presented to a user at a given time exceeds the optimal "seven plus or minus two", a search problem arises in which the user must determine the optimal next choice from

the connection list or "embedded menu" [Shne87].

Labels of links embedded in hypertext are typically words or phrases, and provide little information about their destination. Even worse, some anchors pointing to different destinations use the same words or phrases, for example, "Click here". The process of pausing (either to jot down required information or to decide which way to go) can be very distracting.

Another potential source of cognitive overhead is "user-interface adjustment". Efforts required for this activity may be influenced by various interface features. Examples include the need to move, resize or manually close windows on the screen, and the necessity to switch from one presentation format to another (e.g., from the presentation of contents to the presentation of structure). A number of empirical studies demonstrate effects of such features on various kinds of user performance. For example, several experiments investigating the impact of different window layouts showed that tiled windows - as opposed to overlapping windows - are easier to use and lead to higher accuracy and speed in accomplishing certain tasks [Bly 86].

Considerable research has been undertaken to develop better tools and methods to solve or minimize the problems of disorientation and cognitive overhead. Web browsers are good examples. They make it very easy for users to access Internet information by simply clicking on links. Unfortunately, the Web users still suffer from navigational problems. As the Web becomes increasingly important to business and entertainment and widely used, effective navigation becomes an issue of fundamental importance.

1.4 CZWeb Approach to Aid Navigating the Web

The work described in this thesis addresses the problems of disorientation and cognitive overhead in navigating the Web. We designed a navigation tool (CZWeb) to help Netscape users to navigate through the Web. CZWeb organizes the visited portion of the Web into a map, which enables the user to identify his current position with respect to the overall structure and to easily see the way that led to the current position. The automatically created map may also reduce the user's memory load and

help him to minimize cognitive overhead. The map can be customized and saved for future use. It can also be modified by the user and used to organize information in his own way.

A new approach, URL-based structuring, is proposed for organizing collected information and reducing screen clutter. The structure is displayed on screen with a fisheye-view algorithm which is one of the detail in context approaches used to visualize large hierarchical structures [Dill95]. Extensions to the original algorithm [Dill95] were developed to support both explicit control of node size and maintaining constant size in some given nodes while zooming others.

The rest of the thesis is organized as follows. Chapter 2 discusses some related navigation approaches and systems which were proposed to overcome the hypertext navigation problems. Chapter 3 discusses design goals and gives an overview of features implemented in CZWeb. Chapter 4 describes the implementation details of CZWeb. Chapter 5 summarizes all of the evaluations done so far. Chapter 6 is discussion and conclusion.

Chapter 2

Literature Studies

Since the two most challenging problems, disorientation and cognitive overhead, of using hypertext "may ultimately limit the usefulness of hypertext"[Conk87], many researchers have tried to solve or minimize them. There are a number of research issues related to this. This chapter reviews existing systems developed to address research issues related to these problems. In particular, we focus on an overview diagram approach, which is related to our CZWeb project. Related techniques to display large hierarchical structures are also discussed.

2.1 Designs for Navigation

In a true hypertext system, users must be able to move freely through the system according to their needs, without getting lost either spatially or cognitively. The need for adaptive, intelligent assistance in navigation becomes greater as the complexity of the network increases.

Nielsen [Niel95] describes several possible tools to solve or minimize the navigation problems: *guided tours, footprint, backtrack, history lists, bookmarks, sneak preview, search facilities*, and *overview diagrams.*

2.1.1 Guided tours

A guided tour is a sequence of links designed by the hypertext author that is relevant to a certain topic. Guided tours are most useful for learning systems that provide information on different subjects. But they are difficult to maintain in a changing hypertext like the World Wide Web. Such tours also seem to move us back toward linear text.

2.1.2 Footprints

Footprints provide a visual indicator that a particular node has been visited, an anchor has been activated, or a link has been traversed. Nielsen's system marks anchors with a check mark if they have been activated. Netscape and other web Browsers highlight the labels of visited links.

2.1.3 Backtrack

When you explore a cave, you mark your trail with a thread that can help you find your way back. *Backtrack* is the thread hypertext systems provide for you. It is an important navigation facility, which simply stores the path taken through the hypertext, allowing the user to go back to previously visited pages at any time.

The great advantage of Backtrack is that it reduces the cost of making a wrong decision in choosing a path, and encourages greater exploration.

Nielsen suggests that backtracking mechanisms must fulfill two requirements: "it should always be available, and it should always be activated in the same way." [Niel95]

There are many ways to do backtracking [Niel95]. Netscape's backtrack scheme retains only those pages on the path from the starting page to the current page and all of the other branches are pruned. As a result, users often cannot get back to pages visited just minutes ago, and so get lost during the middle of a navigation session.

For example, if the user started at page A and visited page B, C, D in this order, he is at page D now and he can go back to any of these four pages by using 'Back' or

'Forward' button. If he goes back to B then follows another link to E, all pages after C are discarded and he cannot go to C or D by "Backtrack".

2.1.4 History lists

Textual history is a complete list that shows all nodes (URLs or titles) the user has visited so far, and allows the user to go back to any of them directly. But it is an "unorganized" list of URLs or titles and does not greatly help in locating specific information, because the number of visited nodes becomes very large quickly.

HyperCard [Kaeh88] has a graphical history list called the *Recent list* which has miniature snap-shots of the last forty-two nodes visited. Clicking on a miniature brings that card to the display. This method makes the assumption that a user may not be able to remember the name of the node but may remember the "look" of the node. The *Electronic Document system* [Fein88] implemented a more sophisticated graphical history display. Each miniature is named and time stamped and maintained in exact order. A node visited more than once is repeated on the "Timeline".

Nielsen [Niel95] also referenced two other systems which used miniatures as graphical history list. But some usability studies have shown that it is very difficult for users to distinguish these miniatures from one another [Niel90].

2.1.5 Bookmarks

Tourists take photos of beautiful views and show people where they have been. Bookmarks are like the photos you take in your "hyperspace journey". They provide users with a way of marking key pages so that they can subsequently be quickly retrieved from a list. They are typically used to store useful indices, home pages, and favorite or relevant information pages.

Bookmarks reflect an idiosyncratic and somewhat arbitrary view of the Web, allowing easy access to a small subset of previously visited pages, but not providing any navigational or structural information that can be used to guide searching and browsing strategies to find new and relevant information.

The user has often very little knowledge or remembrance of what the bookmark is about, and how URLs are related to each other. Netscape Navigator 2.0 and later version have added support for tree-like directory (or folder) structure for organizing bookmarks. But the tree has to be managed manually by the users themselves.

Maarek [Maar96] provides a method to automatically manage bookmarks, but it organizes Bookmarks into a binary tree.

2.1.6 Sneak preview

The labels of anchors in hypertext provide little information about their destinations. Some systems, like Hyperties [Shne87], provide a little more information before actually following a link. Search engines also provide "Review" for each of the search results to describe what the destination talks about. This tool helps users to make their decisions.

2.1.7 Search engines

Searching finds information, but sometimes without the thrill and benefits of the browsing-type journey. Various current search tools focus on key words or words embedded in the documents or in the meta-data (associated information). Search engines use "crawler" programs to collect new documents or updated old documents and information retrieval techniques for indexing and storing information to enable fast and accurate retrieval of these documents. They are essentially huge URL databases that hold characteristic information on each URL to determine its relevance to a given query.

Many search engines are available on the Web, such as Excite, Yahoo, Infoseek, Lycos, and Magellan. They are helpful for finding information, but are not robust for use. Thousands of hits with dozens of duplications are returned for a single search, and users have to check all of them to find what they really need.

Nielsen [Niel95] use the term *overview diagram* to refer to a large class of navigational tools which provide visual representations of hypertext structure, such as *graphical browsers* in NoteCards [Hala87] and *map* in Intermedia [Meyr86]. An overview

diagram displays some or all of the hypertext as a graph, providing an important measure of contextual and spatial cues to supplement the user's model of which nodes he is viewing and how they are related to each other and their neighbors in the graph.

2.2 Overview Diagrams

Every hypertext system forms a network of nodes and links, but in many systems that network is only represented inside the computer. At any given time, the user sees only the current node and the links leading out from that node; it is up to the user's imagination to picture how the entire network is structured.

Frank Halasz from Xerox [Hala87] had put forward the view that a true hypertext system should include an explicit representation (a dynamic overview diagram) of the network structure in its user interface. Some hypertext systems did implement this idea.

2.2.1 Global Overview Diagrams

Some systems use one overview diagram to show the entire network, for example Notecards [Hala87] and gIBIS [Conk88]. This kind of overview diagram is called a *global overview diagram.* See Figure 2.1 and Figure 2.2.

Users are able to scroll these overview diagrams as well as rearrange the placement of nodes. Particularly in the gIBIS system, users are encouraged to move new nodes from their default position to make the browser representation more meaningful. Both systems provide a feature for viewing the contents of the browser at multiple levels of detail. If the network is large, the highest level of detail shows the structure of the information, but no semantic information. The user can zoom in to see any portion of the browser in detail, but owing to space limitations, can never see the entire network in detail or in any compacted format that retains semantic information. One useful aspect of these global overview diagrams is that they give the user, at a glance, an idea of the size of the network. Users can tell roughly how many documents they are working with and how interconnected they are.

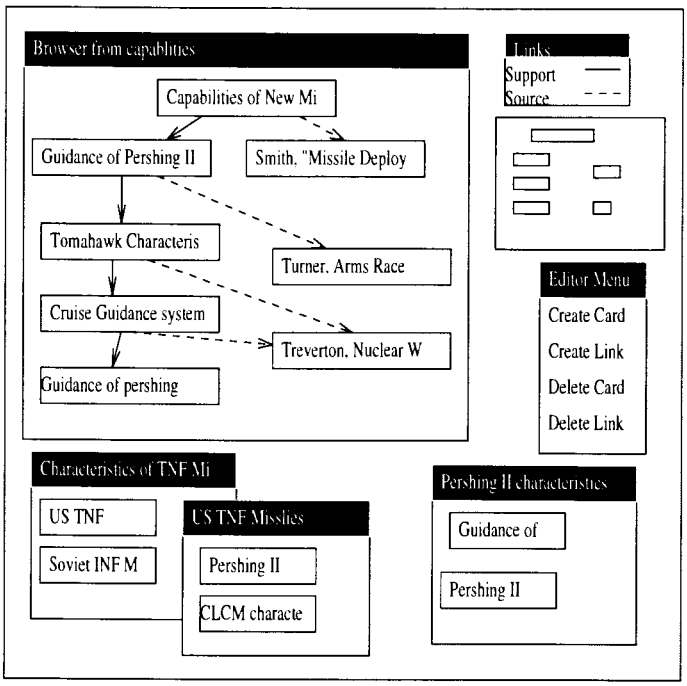

Figure 2.1: Drawing Based on NoteCards [Hala87]

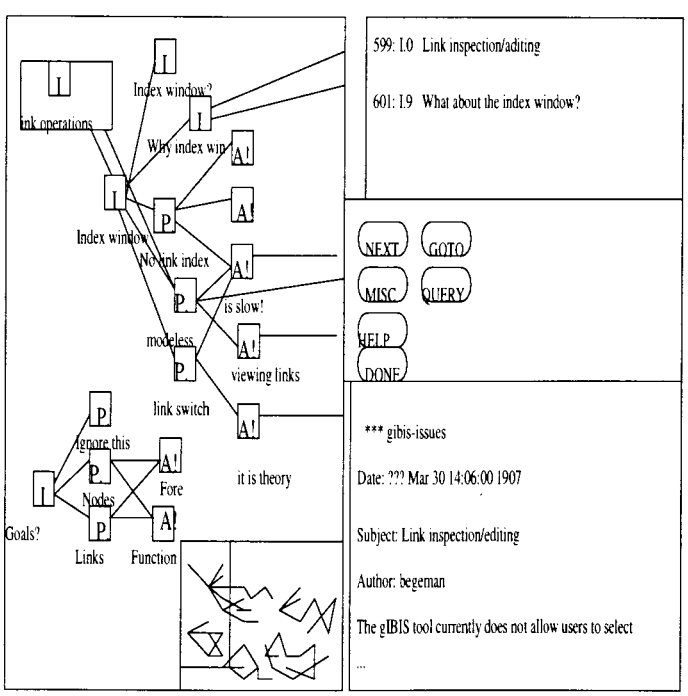

Figure 2.2: Drawing Based on gIBIS [Conk88]

Intermedia [Meyr86] is a hypertext system developed at Brown University. In this system, a *web* is defined as a network of documents or portions of documents linked together. Initially, the designers provided users with three kinds of *Web Views*. The *global map* portrayed every document in the web and the links between them. The *local map* showed a particular "focus" document specified by the user, and the documents to which it was linked. The *local tracking map* was exactly the same as a local map, except that it updated its focus dynamically as the user opened and activated documents.

In a later paper [Utti89], the Intermedia developers found that the global map was not helpful. "The large amount of space required for a complete view of such a web pose a problem for two reasons. First, it is hard for the user to move through such a large space; it requires much mouse movement, there are many patterns and clusters to remember, and so forth. Second, it becomes harder to get all relevant information into the view." They modified the three Web Views into one. The new Web View contains three major components: a *path*, a *map*, and a *scope line*, as illustrated in Figure 2.3.

The new version of Intermedia's map is one way to deal with the sheer quantity of the information in the hypertext. Another solution is using multi-level overview diagrams to show various levels of detail. Jacob Nielsen's system – *HyperTEXT'87 Trip Report* – [Niel90] uses two layers of overview diagrams and displays both of them on the screen at the same time. See Figure 2.4. The *global overview diagram* provides an overall picture and can also serve as anchors for *local overview diagrams*. Local overview diagrams provide a fine-grained picture of the local neighborhood of a node. Both overview diagrams have constant size and the layout of the items in them was hard coded by the author.

For very large hypertext structures, it is an open question whether two levels of overview diagrams will be sufficient. Some approaches to address this problem are discussed below (see section 2.4).

The *Electronic Document System* (EDS) [Fein88] organized "pages" into a hierarchy of "chapters." It consists of two separate components: an authoring system,

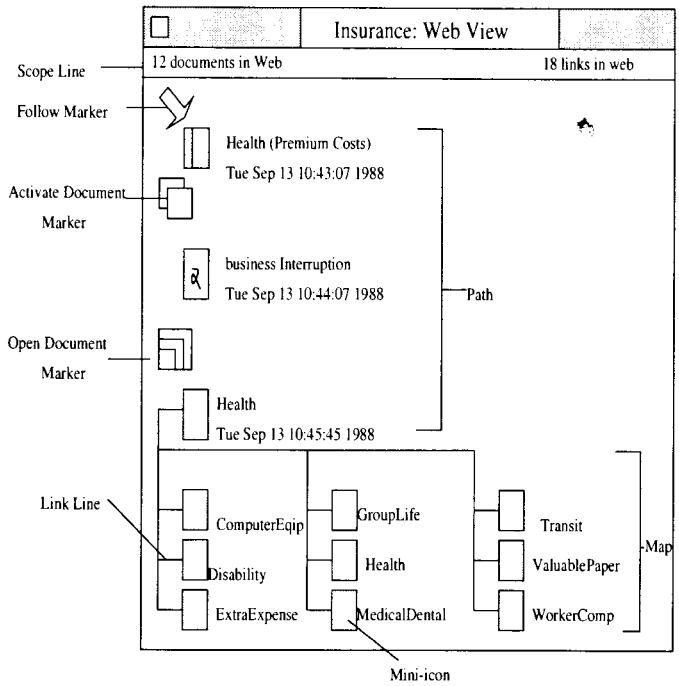

Figure 2.3: Drawing Based on Intermedia Web View [Utti89]

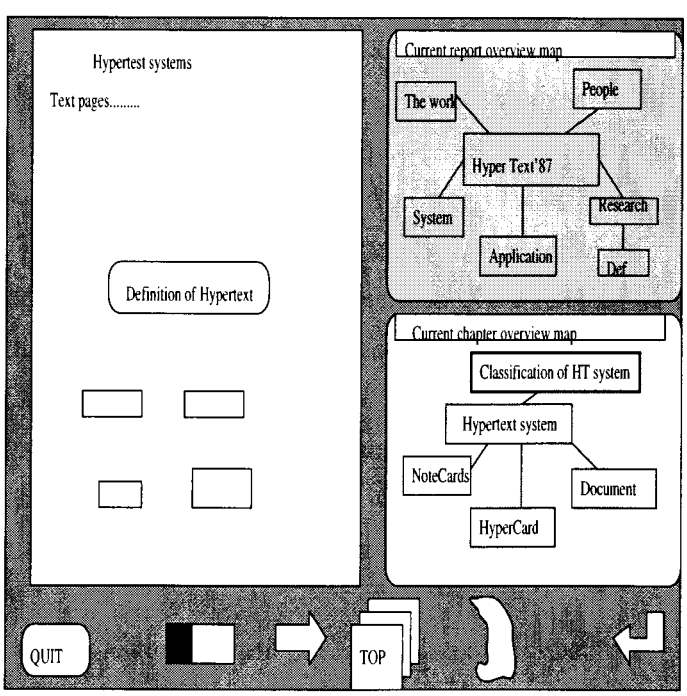

Figure 2.4: Drawing Based on HyperTEXT'87 Trip Report [Niel90]

called *Document Layout System*, and a browsing system, called the *Document Presentation System*. Authors can open multiple windows to view different "chapters" in Document Layout System. It provides a *semi-global* view of the link network, which means that subsets of the data are compacted and represented as a set rather than as individuals. Link can be drawn to or form a chapter rather than the page. Abstracting data in this way makes it more feasible to show connections at the global level.

EDS's Document Presentation System used "Neighbors" to show the network. The Neighbors display is similar to Intermedia's local maps. Because EDS's links are unidirectional, the Neighbors indicate this relationship by displaying the focused node in the center and preceding pages on the left and following pages on the right.

Thoth-II [Coll87] uses a different mapping approach. Rather than building a static overview diagram, it creates an overview diagram dynamically as a user browses through linked nodes. Its "Spider" diagram technique, as shown in Figure 2.5, has a central node and all the other nodes are attached to it directly or indirectly. Each link line is labeled. When the user clicks on an unexpanded node, the diagram expands that node to show all of the nodes that are accessible from it. The expansion creates new instantiations of all the connected nodes, rather than reusing those already exist in the diagrams. Consequently, link lines do not cross each other. However, as the user browses, the diagram quickly gets larger and larger. Tools are provided for scrolling, but this approach is tremendously space intensive.

Foss [Foss88] developed a set of extensions to NoteCards. In her "History List" extension, Foss maintained an ordered list for each NoteCard user. Users can select an item in the list and spawn a "minibrowser". This minibrowser is almost the same as EDS's Neighbors except that footprint were used.

Foss also implemented a "History Tree", which was hierarchical rather than linear. The purpose of this display was to try to give users a sense of how they traversed a set of linked nodes.

Hyper-G [Andr94] is a general purpose, large-scale, distributed hypermedia information system developed at Graz University of Technology. It has a much richer data

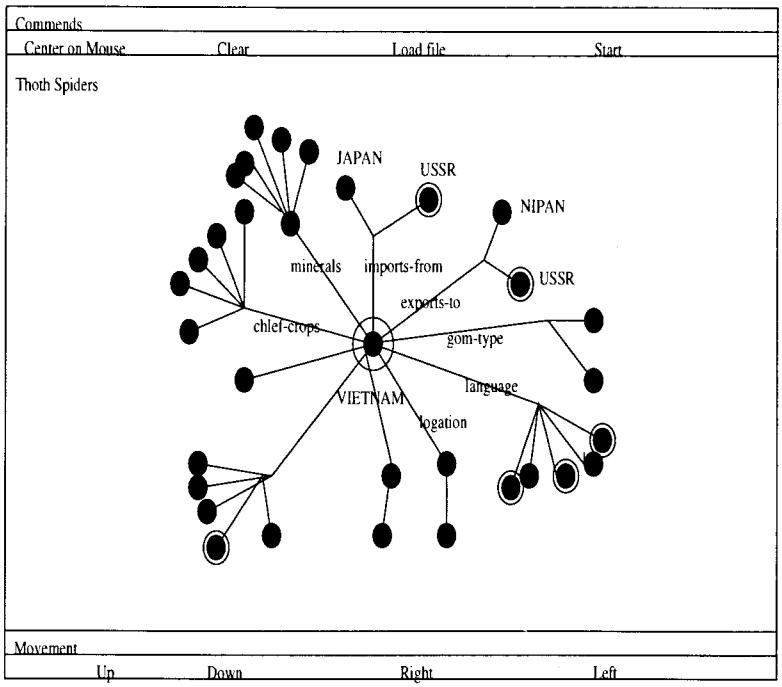

Figure 2.5: Drawing Based on Thoth-II [Coll87]

model on which to base visualizations: a combination of hierarchical structure, (bidirectional) hyperlinks, and fully integrated search and retrieval facilities. Information in Hyper-G may be structured both hierarchically into so-called *collections*, and by means of associative hyperlinks. A special kind of collection called a *cluster* groups logically related or multi-lingual versions of documents. Links in hyper-G are stored in a separate link database and are bi-directional: both the incoming and outgoing hyperlinks of a document are always known and available for visualization.

Harmony is the native Hyper-G client for X Windows on Unix platforms. It takes advantage of Hyper-G's structuring and retrieval features to provide bother intuitive navigation facilities and informative feedback about the location of information. It includes several kinds of dynamic overview and hierarchy maps, a three-dimensional scene viewer, and three-dimensional navigation aids.

The Harmony Session Manager provides navigation through the collection structure, search facilities and various general functions. Central to the design of Harmony is the concept of *location feedback*. The Harmony Local Map facility provides a map of

the hyperlink neighborhood of a chosen document, similar to the local map of Intermedia. However, it can also show other relationships, such as collection membership, annotations, in line images, and the textures applied to a 3D model.

A fundamental problem with built in overview diagrams is that users are condemned to the hypertext system designer's view of the world. If the designer's view does not concur with the users', the overview diagrams the designer created will be of no use to users.

Some hypertext systems, like the World Wide Web, do not provide built-in overview diagrams, but many researchers attempted to build some external overview diagrams to help users to navigate in them.

2.2.2 Overview Diagrams to Aid Web Browsers

The effort of building external overview diagrams to aid Web browsers is focused on transforming the complex underlying Web structure into an appropriate and comprehensible structure. Most approaches try to simplify it into a hierarchy or something similar to a hierarchy.

Basically, there are three kinds of approaches to analyze the hypertext structure: *content-based*, *structure-based*, and *navigation-based*.

Content-based Approaches

In this type of approach, nodes in the hypertext are considered individually and their attributes are examined to determine the structure.

Zizi's SHADOCS document retrieval system [Zizi94] is an example. Zizi defined *Interactive Dynamic Map (IDM)* as a document that provides a global view of either a set of documents or the semantic contents of a large collection of documents. Two types of IDMs are computed from the documents: *Topic IDMs* represent the semantic contents of a set of documents and provide an overview of the topics represented in a collection, their importance, and similarities or correlation among them. It is the counterpart of the global overview diagram in Nielsen's system. The difference is that the semantic information is computed from a web of documents according to

some rule instead of hard coded by the system designer. *Document IDMs* represent collections of documents either generated automatically from a user query or gathered manually by a user or designer. It is the counterpart of local overview diagram in Nielsen's system. SHADOCS implemented three levels of browsing facilities: The wide browsing level uses semantic fisheye views, which is intended for fast scanning over the web of documents; The medium level uses a semantic zoom. The narrow level uses multiple windows, which provide fine-grained views of individual items.

The Document Explorer system [Fowl96] designed at University of Texas analyzes Web documents based on semantic content. It operates on keyword lists to determine associations among documents using a co-occurrence metric to derive similarity measures among documents. Both global structure overview and detailed view are provided.

One problem with this approach is that most of the existing web documents do not contain the required semantic content information. Researchers sometimes have to build test Web sites by inserting some useful information into Web documents.

Structure-based Approaches

Structure-based approaches analyze the system structure based on the link relationship among nodes. They might be described as "exploratory" approaches. They provide the user a condensed visual representation of the structure and content of an information space after analyzing it. The user can evaluate the entire information space at a glance, then focus on individual elements by clicking on objects within the map. Examples of information spaces include Web sites, a collection of Web pages that are unified by a common topic or theme, a whole hypertext system, or a file system. These approaches allow the user to visualize the space without having to visit any documents in the space, but they spend much time on analysis of the structure of the space.

The documents in the information space must be queried as a batch job to determine the structure of documents in the analysis stage. If they are used to analysis the

Web, these batch jobs must be re-run frequently to maintain an accurate representation of the document space because users can continually adding new documents to the server, and the contents of the documents themselves tend to be volatile. This is expensive for both clients and servers.

Rivlin [Rivl94] designed a toolbox which takes a whole hypertext system as input and transfers it into a hierarchical structure. Its hierarchization process solves two problems: identifying a root and distinguishing hierarchical and cross-reference links.

The fundamental property of a root is that every node in the hypertext must be reachable from the root, also, the distance from the root to any other node should not be too large. The author defined and used several metrics to find the root. Once the root is identified. The differentiation between hierarchical and cross-reference links is done with a variation of breadth-first searching to maintain node as close to the root as possible. After forming hierarchical structure, some algorithms are used to break the hypertext into semantic clusters.

NetCarta Corp. [Netc96] has two products: WebMapper which focuses on structure analysis and web site management, and CyberPilot Pro, a navigation tool which helps users find and visualize information. Both are based on the WebMap created after analysis of a web sites.

The WebMap contains two views: Tree View and Cyberbolic View. The Tree View displays the site's hierarchy, much like Windows' File Manager or Explorer. The Cyberbolic view gives a dynamic look at the web site. It shows relationships between pages and other site resources. Both of the views show simplified hierarchical structure to the user and hide the underlying network structure.

The Cyberbolic view always fully expands the whole structure. The view is focused on nodes located near the center. Only those nodes have enough space to be shown in detail. Around the center, nodes may overlap each other but the user can click on a node to bring it to the center and show the details of its neighborhood. The labels of nodes are shown as long and narrow boxes and truncated when there is not enough space. A pop up box with the complete label is shown and remains for a few seconds when the cursor is stationary over an object.

Every object in the map has a picture, or icon, associated with it that gives the

user some hints about what type of object it is. The two views are not duplicates. The user can choose to display different types of objects in each of them. Broken links are highlighted and are very easy to fix with some help applications.

There are some other features such as publishing the WebMap over the Internet, attaching public and private notes to any object, creating reports to show some statistical data about web sites, and a searching facility.

InContext's WebAnalyzer [WebA96] is another commercial available tool for web site management. It has similar features to WebMapper, but the screen is organized differently. WebAnalyzer's screen is divided into three main parts: the *Link View* on the left, the *Wavefront View* on the right, and the *File View* along the bottom. The *Wavefront View* is the global overview diagram that displays each level of a Web site for a complete birds-eye view. Nodes are organized into rings. The starting node is at the center and nodes of the first level (accessible directly from the starting node) form a ring around it; nodes of the second level form a larger ring around the first ring, and so on. It provides the user with a more real structure than WebMapper. But it is very hard to a view large web space with this view.

The Link View is a local overview diagram that displays a selected node at the center and its neighbors around it. This Link View is similar to EDS's Neighbors [Fein88] and Foss' minibrowser [Foss88] except that node are represented by icons rather than rectangles with names inside and in-links and out-links are drawn with different colors.

The File View displays a complete list of attributes of all the files or nodes in the selected Web site. One of the main problems of displaying this kind of structure is that there is no enough space to show labels for each node in the view. Displaying the attributes of nodes (such as nodes' labels) on a separate window makes the two other views which show the structure neatly. But users can not identity an icon in those two views by a glance. They have to select it and check its attributes in the File View.

Navigation-based Approaches

Navigation-based structuring approaches might be described as "reflective" approaches, in that representations are built passively as the user browses the document collections. The application makes no prior assumptions about the structure of the information space and builds the visualization only as new documents are encountered. Thus, the resulting visualization is customized to each session and is built to represent the way the user explores the hypertext.

MosaicG [Ayue95], WebMap [Dome94], and WebJournal [Desa94] are examples in this category. They enhance the history keeping facility of Web browsers by providing a two-dimensional view of the documents a user has visited in a session. By presenting titles, URLs and other attributes of the documents a user has visited, they allow the user to easily recognize previously visited documents and provide easy ways for the user to re-visit those documents and analyze the structure of a set of hypertext documents. Actually, those ideas are similar to Foss' History Tree[Foss88].

MosaicG and WebJournal build their trees based on the hyperlinks embedded in web pages. Documents accessed by following anchors are added as child nodes to the node that represents the document where the source anchor is located. This approach cannot solve the problem of a child node containing a link pointing back to one of its ancestors. In this case, the tree may have an infinite number of levels.

WebMap treats the newly visited documents as child nodes to the current node, which represents the document that is displayed in the Mosaic's top window. This approach does not show the link relationship among the nodes and it depends on users' navigation process only.

MosaicG and WebJournal display their trees horizontally and letting them grow from left to right. WebMap displays its tree vertically and lets it grow from top to bottom. All of the overview diagrams are scrollable.

MosaicG shows users a "perfect" tree where all the cross-reference links are hidden and shown only on request. If a page is the destination of more than one link, a short arrow appears to the left of the node. By positioning the mouse over this arrowhead, the other nodes in the hypertext that contain links to this document are highlighted.

WebJournal uses four different types of links denoted by dashed lines that link nodes outside of the regular tree structure. They represent different actions user actually invoked to get the document, such as clicking on an anchor embedded in the current document, typing in a URL manually (using either the "Open URL" or "Open Local" buttons in Mosaic), selecting a URL from Mosaic's hot-list, and opening a new Mosaic window. The user can prune or expand part of the tree if the size of tree is very large. WebMap defines a new data structure – "spanning tree". Edges in spanning tree are classified into two categories: *Tree-edges*, and *Non-tree-edges*. Different edges are represented by different colors and line patterns.

Nodes have attributes such as URL and title. Besides those, MosaicG also uses thumbnail images to represent nodes, which are similar to miniatures in HyperCard. WebJournal assigns each node a number according to the order of its visit. Numbers are easy to display, because they take less screen space than long strings. But users may not be able to remember which page a number represents.

In addition to the common graphic history keeping facility, WebMap provides a playback feature, which makes it easier to find a previously visited document, and a feature for collection and ordering pages. The author also describes a "domain" concept, which provides a general mean for grouping and structuring HTML pages. A domain is defined as a set of related HTML documents, which together form one document. Traversal strategies are defined for a certain domain structure type. They describe the semantics of several navigation operations that move between the pages of a domain.

The above three approaches have two main disadvantages. The first one is that they modify NCSA's Mosaic browser in order to let it communicate with their applications. Users have to compile and keep a private version of Mosaic. That is not convenient. The other disadvantage is that they make inefficient use of screen space, because they display a visual tree on the screen.

WebViz [Pitk94] is another project based on the users' navigation. It provides Web database maintainers and designers with a graphical view for local databases and accessing patterns. The user can see not only the documents (represented visually as nodes) in his database but also the hyperlinks traveled (represented visually as links)

by other users requesting documents form the database. WebViz further enables user to selectively filter the access log, control bindings to graph attributes, play back the events in the access log, select a layout of nodes and links that best presents the database's structure, and examine the graph at any instant in time.

Hybrid approaches

Each type of the above three approaches has its advantages. Structure-based approaches analyze static data and provide users with an overview of the whole information space. They are often used to design Web site management tools. Navigation-based approaches focus on the part of the information space of interest to the user and provide the user a fine-gained dynamic map. They are used to design navigation tools. Content-based approaches help users find related information from the information space and can be used in both management and navigation tools. Actually, many projects combine these approaches or construct different overview diagrams based on different approaches. Neuwirth et al. [Neuw87] observed that the ability to view knowledge from different perspectives is important to help better comprehend the information. The following are some of the hybrid approaches.

HyperSpace [Hend95] has two components, one is *structure-based* while the other is *navigation-based.* The *structure-based* component reads in a list of web pages and presents the user with a three-dimensional structure representing the relationships among them. Instead of enforcing a predetermined arrangement on the display of the graph as other visualization tools attempt to do, HyperSpace allows the set of nodes and arcs to evolve its own form. This self-organization is done by randomly placing the nodes in three-dimensional space and allowing a set of forces to act on them until equilibrium is reached. Two types of forces are used: a repulsive force between all objects and an attractive one between objects that are related (connected by a hypertext link). The system of forces pushes all completely unrelated objects apart, but keeps linked pages nearby, creating an emergent structure where pages linked to similar things are brought close together. HyperSpace provides several useful features to help manage large structure. Arcs between nodes may be removed and

only shown on selected objects when needed, in case they will obscure useful structural information. Virtual reality techniques are used and allow the user to move around in the three-dimensional space to view the details of any parts of the space.

The *navigation-based* component is similar to MosaicG, but it builds a 3D map and supports collaborative browsing. Rather than just observing a single user's navigation, the system would watch several users' progress through the web and dynamically collate the information into a database representing the group's experience of the web. Each individual would have a separate view of this database in HyperSpace, and would be able to customize it to his own particular preferences and needs, allowing him to independently examine and explore any part of the web. However, the data collected and represented would reflect the group's overall progress and concerns. This technique can introduce users to parts of the web that they have not themselves explored but that have been mapped by others. In the collaborative system, new areas are presented to the user with a ready built framework to navigate through, instead of the blank unknown of completely unexplored links. This is the only project, I am aware of, that supports collaborative browsing.

Hy+ [Hasa95] integrates structure-based and navigation-based techniques together. Basically, it is based on navigation. When the user visits a web page, all of the web documents that are accessible from this page are displayed on the right to this page and linked with "ntr" edges. Web documents are represented by icons with their URLs displayed to their right. Followed links are represented as "tr" edges, which shows the user's navigation path. See Figure 2.6 for details. It is similar to Thoth-II. But Hy+ reuses existing icons instead of creating a new one each time and also provides filter facilities.

The *GraphLog* query language can be used to restrict the set of documents displayed in the view, using flexible criteria, including structural properties and regular expression matching on URLs, anchor labels, and document titles. Another advantage of this view is that the user has easy access to any documents which are referenced within some visited documents without having to fetch these documents again. Its main disadvantage is the inefficient use of screen space.

Mukherjea's *Navigational View Builder* [Muk195, Muk295] provides the user with

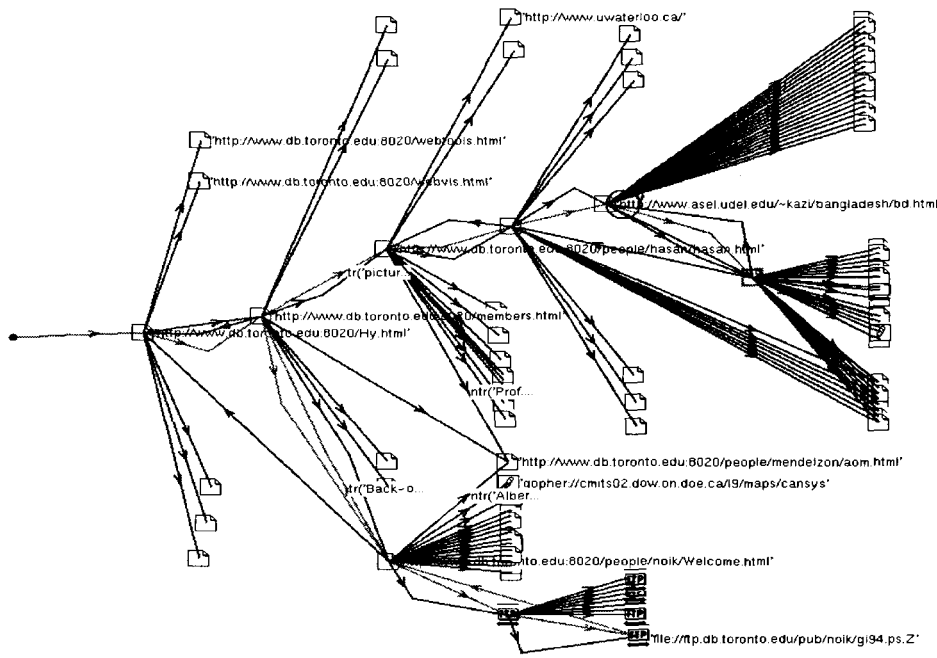

Figure 2.6: Graphical view generated by Hy+. Printed with permission.

different hierarchies, each giving a different perspective to the underlying information space and claims that it would help users to comprehend the information space better. The hierarchies are formed automatically from hypermedia networks based on content and structural analysis. The structure they get is called *pre-tree*, which is an intermediate between a graph and a tree. A *pre-tree* has a root but its descendants may be graphs, which are called *branches.*

The structural analysis looks at the structure of the graph and forms pre-trees. The *branches* of a pre-tree are formed with content analysis based on the attribute values of nodes. The user can guide the process to form different pre-trees that give different perspectives to the underlying information. These hierarchies can be visualized in different ways. Once a hierarchy is formed from the original graph structure, the hierarchy can be transformed to other data organization as well. Visualizations can be formed for these data organizations also.

After transferring the information space into a pre-tree structure with many layers

of abstractions, *Navigational View Builder* builds a three dimensional space to visualize it, which combines global and local overview diagrams together. Different layers are located at different distances from the viewer, while the most detailed layer at the near and the most abstract layer at the far end.

One of the disadvantages is that information is lost. Some cross-reference links have to be removed in the formation of pre-tree. The other one is that the web documents do not contain much useful information for content-based structuring. The authors have to insert some information manually to their test Web site. If it is used to visualize other real Web sites, it may not be able to achieve the expected results.

The *Enhanced Mosaic* designed by Gershon [Gers95] implemented three functions. Its *Hyperspace View* uses structure-based approach to analyze hyperspace structure and depicts the result as a visual "tree". This is similar to Hy+, but its tree can grow automatically up to a specified number of levels instead of one level.

The second function attempts to overcome the rigidity of the Web by allowing the user to construct interactively and visually a personal hyperspace of information which links the documents according to the application or problem domain, or to the user's own perception, experience, culture, or way of thinking. The smallest unit of information on the Web is a document (or HTML "page"). But Gershon's Mosaic allows users to define new documents that contain fragments of existing documents and link them to other documents as they wish.

The third function includes discovery and analysis of new information and relationships in retrieved documents by aggregating relevant information and representing it visually. It is a content-based approach.

In this sub section, we focused on approaches helping navigation in the Web based on exist web browsers. This type of overview diagrams is used to show the Web structure only. Web pages are represented as nodes in overview diagrams. The contents of Web pages are still shown in Web browser's window. One of the disadvantages of this type of approaches is that two applications are running at the same time, the web browser and the application providing the overview diagram. These two applications compete for screen space and users have to switch focus between them. There are

some other approaches which explore mechanisms of combining them together into one application.

2.2.3 Overview Diagrams in New Web Browsers

One of the limitations of currently popular Web browsers is that only one Web page can be shown at a time. Some researchers have explored the possibility of building new Web browsers showing multiple web pages at the same time [Brow95] [Card96] [Bede97] [Lamp95]. Each web page can be scaled to a readable size and links on it can be activated to retrieve other web pages. There are also overview diagrams to show the structure of these displayed web pages. In this way, functions of Web browsers and overview diagrams are seamlessly combined into one application. Following are a few examples.

DeckScape [Brow95] is one of the new browsers that changes what is currently a "standard" depth-first paradigm into a multi-level one. DeckScape uses a *deck*, a collection of Web pages of which only the top one is visible at any time, to organize Web pages. Each "deck" is traversed linearly, but separate decks may be maintained in parallel, and documents may be moved between decks. With this approach users have more control over navigating a set of disjoint pages that are either unrelated or branch out from a common ancestor. Users can switch back and forth among unrelated pages easily. They can use decks to help find pages that they have visited before. Decks can also be used to organize hotlists or used to contain returns of results of certain operations. More importantly, unlike the common browsers, state (the contents of a deck) is preserved across invocations, so users can retain context over time.

Books have been the most prevalent sources of information for hundreds of years. They are common media to convey information. Using books to organize pages is more acceptable than using decks. WebBook [Card96] implements this idea. It arranges related single Web pages as a higher aggregate entity with the physical book metaphor and allows rapid local interaction with it. This lets users do the most elementary operation of sense-making: grouping. Instead of waiting for each page on a bookmark

to be accessed, users can access pages immediately, even fan them for rapid scanning. The embedded Document Lens [Robe93] can be used to inspect portions of interest. The user is able to pan and zoom over the entire set of pages, while retaining a focus plus context display of the book.

Given a collection of web pages, WebBook pre-loads those pages and displays them as a collection using an augmented simulation of a physical book. 3D graphics and interactive animation are used to give the user a clear indication of the relationship between the pages of the book. Links are color coded so the user can easily tell the difference between a reference to another page in the book and a reference outside the book. The WebBook takes advantage of advances in graphics and processor power to get much closer to a realistic simulation of a book. At the same time, it goes beyond what is possible with a physical book.

DeckScape and WebBook focus on organizing Web pages based on the content of web pages. At a given time, only the page on top of a "book" or "deck" is visible. On the other hand, the pad++ zooming browser [Bede97] and Lamping's hyperbolic browser [Lamp95] focus on showing the link structure among Web pages. They show Web pages with different scale and show links among them.

The pad++ zooming browser depicts Web pages on a large "zoomable" information surface using Pad++ [Bede94], a substrate for building "multiscale" dynamic user interfaces. Pad++ provides an extensive graphical workspace where dynamic objects can be placed at any position and at any scale. In the pad++ browser, only the Web page in focus is scaled to readable size and other Web pages are shown at smaller scales to provide context. As a link is followed, a new page becomes the focus and existing pages are dynamically repositioned and scaled. Layout changes are animated so that the focus page moves smoothly to the center of the display surface while contextual information provided by linked pages scales down.

The hyperbolic browser [Lamp95] uses a hyperbolic geometry to display the structure and control size and location of nodes in the view. It displays a tree with its root at the center, but the display can be smoothly transformed to bring other nodes into focus. In all case, the amount of space available to a node falls off as a continuous function of its distance in the tree from the point in the center.

This section has reviewed different overview diagrams used to help navigation in hypertext. The next section discusses approaches to displaying large information spaces on screen.

2.3 Detail-in-Context

Viewing very large information spaces is a challenging task. The conventional display approach maps all of the information into a region that is larger than the display and then uses scrolling to move around the region. This approach has the problem that the user cannot see the relationship of the visible portion to the entire structure (without auxiliary views).

It would be useful to be able to see the entire hierarchy while focusing on any particular part so that the relationship of parts to the whole can be seen and the focus can be moved to other parts in a smooth and continuous way.

Approaches that display some portion of the information at a greater level of detail while still displaying all or much of the context (the overall structure) are known as *detail in context* techniques. Several such techniques have been developed to address the needs of many types of information structures. Many of these techniques could be applied to browsing trees laid out using conventional 2D layout techniques.

Fisheye views [Furn86] show the entire information space in a single view using varying levels of detail. It is based on the fisheye lens metaphor where objects in the center of the view are magnified and shown with great detail. Objects farther away from the center are gradually reduced in size and detail. Furnas also suggested a "degree of interest" (DOI) function which assigns a value to each node in accordance with the degree to which the user is interested in seeing that node.

The use of fisheye views requires two properties of the information space: it should be possible to estimate the distance between a given location and the user's current focus of interest, and it should be possible to display the information at several levels of detail. Both conditions are met for hierarchical structures.

The *Continuous Zoom algorithm* [Dill95] is a fisheye view technique suitable for interactively displaying hierarchically organized, two-dimensional networks. Nodes

are displayed as rectangles. A parent-child relationship among nodes is shown with a containment relationship, where the container is the parent. Links between nodes can be used to represent other relationships among nodes, such as a network. Nodes can be displayed with different levels of details. Users can control detail displayed by expanding and shrinking nodes.

The *ShriMP* (Simple Hierarchical Multi-Perspective) layout adjustment algorithm [Stor95] is modified from the Continuous Zoom algorithm. It uniformly resizes nodes when requests for more screen space are made and preserves straightness of lines and graph topology in the adjusted views. The algorithm is flexible in its distortion technique and can be changed to suit different graph layouts. It has been used in the Rigi system [Stor97], a reverse engineering system designed to analyze, summarize, and document the structure of large software systems. It helps visualizing and navigating software structure modeled as nested graphs. The fisheye view in ShriMP provides a mechanism to zoom into source code while retaining the context of the software architecture.

3D Pliable Surface [Carp95] is a technique for displaying some large information space, such as graphs or maps, by providing distorted views. It can select arbitrarily-shaped regions (foci) on the surface and control the level of detail of display by pulling them towards or pushing them away from the viewer. Multiple foci are smoothly blended together such that there is no loss of context. The manipulation and blending of foci is accomplished using a Gaussian curve-based mathematical model.

Cone Trees [Robe91], developed by researchers at Xerox PARC, is one of the earliest instances of information visualization of hierarchical structures. A cone tree is a 3D representation of hierarchical information: any one node of the tree is located at the apex of a cone and all of its children (the information subordinate to that represented by the node) are arranged around the circular base of another cone. In contrast to node and link trees, the cone tree makes the largest amount of information visible at same time. Any node can be brought to the front by clicking the mouse on it and smoothly rotating the tree (which also helps the users to keep the larger structure in mind).

Tree maps [Shne92] is another technique for visualizing hierarchical data. A hierarchy is drawn as a set of nested boxes in which each node is depicted as a rectangular region composed of the rectangular regions that represent its children. The main advantage is its ability to visualize large hierarchies, which it achieves through its linear top-down space-filling layout algorithm. It can emphasize nodes by "weight". The disadvantage is that hierarchical structure is not clear.

Lamping's *Hyperbolic Geometry* method [Lamp95] lays out the hierarchy in a uniform way on a hyperbolic plane and maps this plane onto a circular display region. This supports a smooth blending between focus and context, as well as continuous redirection of the focus.

The hyperbolic plane is a non-Euclidean geometry in which parallel lines diverge away from each other. This leads to the convenient property that the circumference of a circle on the hyperbolic plane grows exponentially with its radius, which means that exponentially more space is available with increasing distance. Thus hierarchies - which tend to expand exponentially with depth - can be laid out in hyperbolic space in a uniform way, so that the distance (as measured in the hyperbolic geometry) between parents, children, and siblings is approximately the same everywhere in the hierarchy.

While the hyperbolic plane is a mathematical object, it can be mapped in a natural way onto the unit disk, which provides a means for displaying it on an ordinary (Euclidean) display. This mapping displays portions of the plane near the origin using more space than other portions of the plane. Very remote parts of the hyperbolic plane get miniscule amounts of space near the edge of the disk. Translating the hierarchy on the hyperbolic plane provides a mechanism for controlling which portion of the structure receives the most space without compromising the illusion of viewing the entire hyperbolic plane. The author provides effective procedures for manipulating the focus by using pointer dragging and for smoothly animating transitions across such manipulation.

HotSauce (or *Project X*) [Proj96] is a 3D browser that accepts MCF (Meta Content Format) files. It presents content in a view with floating nodes. Users can fly through the space in 3D with mouse or keypad, quickly navigating through branching

structures. Siblings of nodes are grouped together such that the user can distinguish each of them when close enough. This technology prototype will allow users to move forward, backward and laterally within the HotSauce window. The content of the window is called the *X space*.

Pad++ [Bede94] provides a 2D Pan/Zoom space which is a powerful visualization tool for hierarchical data. The Pad++ surface is an infinite "zoomable" space where objects have an absolute physical location. The metaphor is not one of viewing a small part of the data through a window, as in MS-Windows. Instead, all of the data is placed directly on the Pad++ surface, and users navigate through the data by panning and zooming.

In Pad++ there is a sense of peripheral awareness. When objects are small or off to one side, the user can still see them, and can still make out a little of their contents, though they are not shown in all their detail. When the user looks directly at them and zoom in, all of the detail becomes available. Hence the Pad++ metaphor offers a new route for tapping into our natural spatial and geographic ways of thinking.

We have reviewed several systems and approaches, designed for orientation and reducing cognitive overhead. Overview diagrams are widely used to help users to understand hypertext structure. Detail-in context technique is an appropriate technique to display large hierarchical structures. The Continuous Zoom algorithm has been successfully used to visualize large networks and software structures, but it has not been used to visualize the Web structure. This is what CZWeb actually does. It uses Continuous Zoom algorithm to display a simple view showing Web structure built based on navigation and attempts to help users to navigate the Web. In the next chapter, we shall discuss CZWeb's design goals and its features.

Chapter 3

CZWeb Design and Features

3.1 Design Goal and Challenge

The Web is currently the most popular hypertext system. Though graphical browsers such as Netscape, Mosaic and Internet Explorer implement many navigation tools for the user such as Footprints, Backtracking, History List, Bookmarks, and Search engines, navigation problems are still present. More adaptive, intelligent assistance in navigation is needed.

Our goal is to provide Netscape users with aids intended to help reduce the problems of disorientation and cognitive overhead.

Overview diagrams are one of the good tools to provide contextual information, which is helpful for navigation in hypertext. But Netscape and other web browsers do not provide overview diagrams, because of the sheer quantity of information and fast changing nature of information on the Web. The logical choice is to build some external overview diagrams. However, constructing effective overview diagrams for the Web is a challenging task. There are four main problems involved:

- The size of the Web is so large that it is hard to provide so much information in one overview diagram. Even if such an overview diagram were implemented, users may not need it. Each user may have his/her own interest in particular web sites or information related to a particular topic. Nobody has the time

to view all of the documents on the Web. The problem we must solve is how to filter the information available on the web and provide only important and useful information to the user.

- The Web is a very complex multidimensional network whereas traditional display techniques are only two and three dimensional. How to project such multidimensional data onto a two or three dimensional display without losing information is also a problem.

- After deciding what to present, the next problem is how to show it on the screen. The size of the screen is limited and it is hard to fit the whole information structure on it. It would be useful to be able to see the entire information space while focusing on any particular part of the space so that the relation of parts to the whole can be seen(detail in context).

- The user interface should allow the user to easily view and manipulate the information displayed. Web browsers are designed for general web users rather than computer experts. Anyone who can use a mouse can use a Web browser. Overview diagrams used to assist Web browser users should have a user interface consistent with Web browsers.

In this chapter, I will discuss our approach to the above problems and certain design aspects of their integration into CZWeb.

3.2 What Information is the Most Important?

Because of the size of the Web, it is not a practical idea to build a simple overview diagram containing all of the information in this hyperspace. All of the approaches and systems designed to study the Web visualize only part of the hyperspace. The structure-based approaches and content-based approaches build overview diagrams to show the structure of local Web sites. The navigation-based approaches organize only visited part of the Web.

The goal of some Netscape users is to find and assimilate the information on the Web. We believe that the historical information is more important than the whole structure of a Web site for this group of users. This is based on the methods people use in searching for information on the Web. Historical information is defined as the temporal sequences of page visits and the internal hyperlinked structure of those visited pages.

When people search for information on the Web, three possible methods are:

- Use search engines available in Netscape. These search engines will return a long list of items.
- Start from a web page that contains links to other web sites or documents. The URL of the starting page may come from another person, or might be discovered by accident.
- Browse a Web site from the home page of a person or an institution.

For the first two methods, hyperlinks in the starting pages point to different Web sites located around the world. If we use structure-based approaches to map all of these web sites, it may take a very long time and create a very complex structure.

For the third method, we can use structure-based approaches to map a local Web site. But, even if we get a map of the structure of all the web documents, it is still very hard to find relevant information, because Web does not provide useful semantic attributes, such as topics, for documents. Actually, a Web site may contain a very large number of pages, most of which are not related to what is sought.

In all three cases, browsing through hyperspace is necessary. Navigation-based approaches, which provide maps to show the structure and relationships of all visited Web pages would seem most useful.

3.3 Transforming the Web Structure

3.3.1 Mental Models of the Web

It is generally assumed that users apply *mental models* in trying to understand and predict system behavior, and that their success in using a system depends on how well their mental models correspond to a model represented in the system design [Baec95].

The underlying structure of the Web is a network. In order to better understand users' mental models of the Web, an informal survey was done as part of the CZWeb usability study project for an HCI course[Cour96]. We showed our subjects four diagrams, see Figure 3.1, and asked them to select one that best represents the Web. Four of the five subjects selected the second diagram.

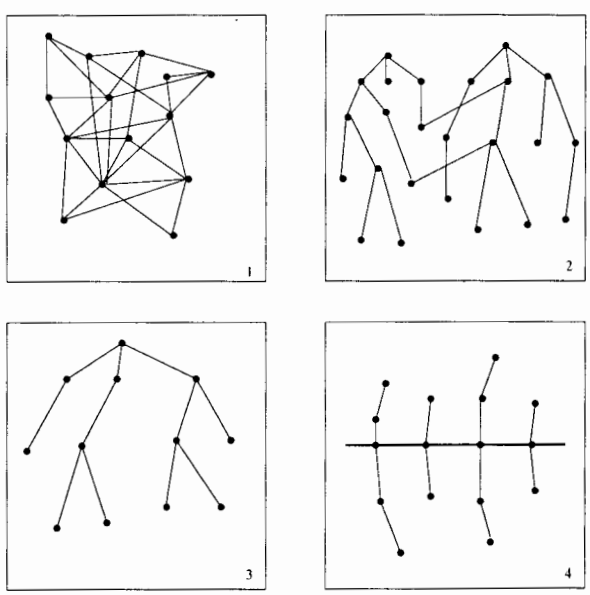

Figure 3.1: Diagram Used in Interview

Based on our interviews, our subjects thought that the Web consists of many connected sites, separated in space, and that the information was stored in a hierarchical structure like a file system at each site.

Carroll [Carr84] pointed out that "People will map a new concept to what they already know, which may help them to understand the new concept." The Web is

a new concept to users, though it is more complicated and less organized than a file system, but people are familiar with file systems, so that it seems natural for them to think of the Web as a big file system.

3.3.2 Managing Historical Information

We also designed a "treasure hunting" game to help in our CZWeb usability study. The subjects were given a starting Web site with some clues and were asked to find a special page containing some required information. In order to finish this task, they should understand how information was organized in that Web site. Unfortunately, Netscape does not provide this kind of information automatically. They had to find it out for themselves by browsing. The design of the game was not successful. Among the five subjects we had, only one subject found two of the three "treasures", three subjects found one and one subject found nothing. However I found something interesting by observing the subjects' searching patterns. Starting from the given page, they selected a link and followed it. If they could not find the information there, they might go one more step deeper. If they still could not find what they were searching for, they would, in most cases, use the 'Back' button once or twice and go back to previous pages and try another path. In other words, they kept their navigation within three levels most of the time. Actually, some "treasures" in this game were located deeper than three levels. This was the main reason why the subjects could not find them.

Catledge and Pitkow [Catl95] have obtained a similar result from their strategies studies: "... The example above is very typical in that users rarely traverse more than two layers in the hypertext structure before returning to an entry point. Initial evidence suggests that this pattern occurs independent of hyperlink per page ratios."

When users try to find some information about an institute, they usually start from its home page. The reasons are:

- All information about a institute are accessible from its home page
- URLs of home pages are easy to remember

- Many users save home pages in their bookmarks
- URLs of most of the home pages can be coined from the name of institutes

Searching information starting from home pages is much like our "treasure hunting" game. Web documents of an institute are arranged hierarchically. After visiting the home page, which is located at the top level in the hierarchy, users traverse down the hierarchy to read one subject in depth, and then backtrack up to the top level to find another subject.

The facts we described above suggest that a user may be able to manage a three-level tree in his mind without external aids. Based on this suggestion, if we can build a hierarchy similar to the one in the user's mind but with more levels of detail and show it visually, users may be able to combine these two hierarchies easily. On the other hand, linear history lists provided by Web browsers are hard to transfer into a hierarchy.

This suggests that navigational aids to web browsers should display not only the sequence of web page visits but also the underlying hierarchical structure. The challenging task of showing Web structure is transferring it into an appropriate structure which can be displayed on screen. Link-base structuring is used in many approaches.

3.3.3 Link-based Structuring

Many approaches have been developed to organize the historical information into a visual tree or tree-like structure, such as MosaicG[Ayue95], WebMap[Dome94], WebJournal[Desa94], and Hy+[Hasa95]. Visited documents are represented as nodes and hyperlinks are represented as edges in these trees.

The tree structures they build are based on the link relationship embedded in web documents. When they simplify the underlying network structure of hypertext into a tree or tree-like structure, a node is considered to be a child node of another node if there is a hyperlink pointing from the latter to the former. I call this type of approach "link-based structuring" approach.

The main disadvantage of link-based structuring approaches is that some information is lost during the structuring process. The World Wide Web is a graph or network,

which means there can be many cross-connections between web documents. Those cross-connections are very important for understanding the web structure. But tree structures cannot contain cross-connections. In order to simplify the web structure into a tree, two approaches have been discussed in the literature. One is represented by Thoth-II [Coll87] and the Tree View of NetCarta's WebMapper. Here, nodes are not reused and new tree nodes are created when needed so that a node in the hypertext may be represented with several nodes in different places in the tree. If node A contains a link to node B and node B contains a link back to node A, the tree will have infinitely many levels! Another disadvantage with this approach was that the use of space was inefficient.

The other approach is used more often, and simply removes or hides those cross-connections. Navigational View Builder [Muk195] removes part of the cross-connections and builds a "pre-tree". The Cyberbolic view of NetCarta's WebMapper and MosaicG keep the cross-connections internally but hide this information from the user. From the structure built in this way, the user may get the mistaken idea that the Web structure is just a hierarchical structure.

3.3.4 Our Approach – URL-based Structuring

We propose a novel approach to represent the web structure. Our structure preserves the cross-connections of the Web explicitly *and* shows the correct network structure to the user.

Since the Web structure is a network, representing it with a network will not lose information. Pages are represented as nodes and hyperlinks are represented as links in the network. As we pointed out in the previous chapter, the main disadvantage with this approach is that the screen gets cluttered with too much information.

Our solution to the cluttering problem is organizing the nodes into a hierarchical tree. With this abstraction, it is easy to control the amount of information to display. Thus there are two structures in the same overview diagram, a network and a tree. The network is used to represent the hyperlink relationship among nodes and the tree is used to organize the nodes. Nodes are shared by both the network structure and

the tree structure.

In our overview diagram, the network structure is shown using arrows to represent links and the tree structure is represented by containment relationship among the nodes.

An algorithm I developed to build the tree is based only on the *structure* of the URLs of the web documents, so that I call it a *URL-based structuring* approach. The relationships among web documents (i.e. the hyperlinks embedded in them) form the links in the network; we note that they have no contribution to building the tree. A user's actions in accessing web pages (activating anchors, typing in URLs or selecting items from history list or bookmarks) will affect the network structure, but will not affect the tree structure. This is the main difference between URL-based structuring approach and other link-based structuring approaches.

Before discussing the URL-based structuring approach, we briefly review the structure of URLs.

3.3.5 The URL Structure

A URL (Uniform Resource Locator) is an address for a piece of information on the Web. It is unique for every item, and every item has a URL. A URL is composed of three pieces of descriptive information about a document:

- The protocol which is used to speak with the server on which the item resides
- The Internet name of the server
- The file name of the information item on the server

These three pieces of information are put together in a standard to form the URL:

protocol://servername/filename

For example, URL **http : //fas.sfu.ca/cs/research/groups/Graphics.html** refers to a document describing our graphics laboratory. This document is available using the HTTP protocol. The server on which it resides is called **fas.sfu.ca**, the

server for the School of Applied Science in Simon Fraser University. The file name is **/cs/research/groups/Graphics.html**.

The servernames are organized into a hierarchy using a technique called the *Domain Name System*. The Domain Name System is a method of administering names by giving different groups responsibility for subsets of the names. Each level in this system is called a *domain*. The domains are separated by periods [Krol95]. As you proceed left to right through a servername, each domain you encounter is larger than the previous one.

Files residing on a server are organized into a hierarchy too. This hierarchy is the same as a UNIX file system. Each level in the file system is called a *directory*. The directories are separated by "/". The container relationship order is reversed from that of the Domain Name System. In the name **/cs/research/groups/Graphics.html**, file **Graphics.html** resides in directory **Groups**, which is a subdirectory of **research**. Directory **research** is a subdirectory of **cs**.

Cluster-Page Structure

In our first approach [Coll95, Coll96], we automatically organized nodes into a hierarchy with two levels. Two types of nodes, *page nodes and cluster nodes*, are used. Visited web pages are represented as *page nodes*. When the user visits a new page, CZWeb creates a new page node to represent it. Page nodes are grouped with *cluster nodes*. We retrieve the server names from the URLs of the pages and create a cluster node for each server. All of the pages from the same server reside initially in the same cluster node representing that server.

Links are drawn from page node to page node. When the user activates a hyperlink in the current page displayed in Netscape's window and visits a new page, a link is created from the current page to the new page, if one does not exist.

This approach helps alleviate the cluttering problem, but does not solve it. If the user visits many pages located in the same server, the cluster node representing that web server will be very cluttered.

If the user visits many servers, the top-level window will be cluttered. In both

cases, the user has to manage the screen space manually. CZWeb allows the user to modify the automatically created hierarchical structure by dragging a node and dropping it into another node.

URL-tree Structure

In order to solve the cluttering problem, a more sophisticated approach is needed. I propose a new approach called *URL-tree structure*. With this approach, there is only one type of node and all of the nodes are organized into a tree, called a *URL-tree*. The URL-tree is created based on the structure of URLs of web pages which the nodes represent and can have any number of levels. Nodes in the URL-tree may represent Web pages or Web servers. The URL-tree structure is an extension to the Cluster-page structure; it is a more abstract representation of the information and it provides an additional stage of information abstraction. A detailed description of creating the URL-tree will be postponed to the next chapter.

3.4 Visual Representation of the URL-Tree Structure

Some navigation-based approaches draw trees or graphs on the screen to represent their "tree" structures. These approaches have two disadvantages: first, the screen space is not used efficiently; second, when adding a new node or deleting an existing node, all of the nodes and edges in the tree have to be reorganized in order to create an aesthetic layout of the tree. The user may lose any sense of structural continuity with this kind of continuous change.

CZWeb provides a single overview diagram to show its internal structure and it uses the Continuous Zoom technique to manage the display space. With the Continuous Zoom algorithm, the tree structure is represented by containment relationship among nodes rather than by edges. Nodes are represented as rectangles with their children inside. The user can interactively and continuously change both the size and location of any node. When the size of a node is changed, the size of other nodes will

be changed accordingly to use available display area effectively and to avoid nodes overlapping. The algorithm provides a smooth change in visual state, which gives the user a sense of visual continuity among states. The topological relationship between nodes is also retained after the change. When new nodes are added in, the structure is changed. It may influence the location and size of existing node. We are seeking algorithm to minimize the influence.

Another key advantage of the Continuous Zoom algorithm is that it supports multiple focus points – more than one node at different parts of the overview diagram can be focused on and opened to varying levels of details at the same time.

Nodes managed with the Continuous Zoom algorithm have two states: opened and closed. Opened nodes show their children while closed nodes hide their children so that they take less space. The user may close or open nodes to control the space and visual clutter explicitly. In this way, the user may focus on a part of the overview diagram to look at any desired level of detail while still retaining the surrounding context.

In CZWeb's implementation of the Continuous Zoom algorithm, cluster nodes can be opened and closed. Page nodes only have one state. They cannot be closed but are resizable. Because page nodes are leaves in the hierarchy and they do not contain other nodes.

The graphical representation of a page node is a rectangle with its label displayed inside. The user can choose to display any one of its three labels: URL, title, or a name assigned by the user. There are two control handles available for the user to control the size (the rectangular area) of the node. The one at the upper right corner is the *zoom* handle. When the zoom handle is dragged, the Continuous Zoom algorithm is applied and it will adjust the size of all nodes. The handle located at the lower right corner is called the *resize* handle. Dragging the resize handle of a node only changes the size of the node itself.

An opened cluster node looks like a window. In addition to the zoom and resize handles, it has a *title bar* and a *close* box. The title bar displays the name of the cluster node. The close box closes the cluster node. A closed cluster node is displayed as a folder icon with its name underneath (and all of its children hidden)(See Figure 3.2).

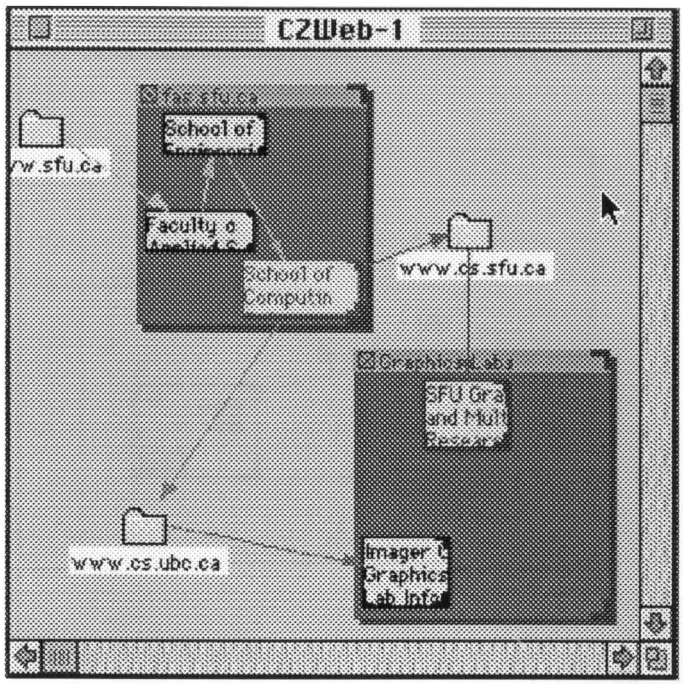

Figure 3.2: A Typical Overview of Cluster-Page Structure

Links are represented as arrows between two nodes. An arrow pointing from page node A and to page node B means that the user followed a hyperlink in the web page represented by page node A and went to the web page represented by page node B. Links are visible if both of the linked nodes are visible. If a node is not visible (one of its ancestors is closed), all of the links that link to or from it are instead linked to or from its nearest visible ancestor. All of the links inside a closed node are hidden. With this abstraction, we can reduce the cluttering of links.

The overview diagram not only displays the structure, but also is interactive and **provides a facility to access any visited web pages easily.** Nodes in the overview diagram form patterns. The user can easily recognize visited pages from those patterns and their locations. Double clicking on any page node brings Netscape back to the web page associated with that page node. Double clicking on a closed cluster node opens that cluster node.

The page node representing the current web page displayed on Netscape is always highlighted and serves as a reference for orientation.

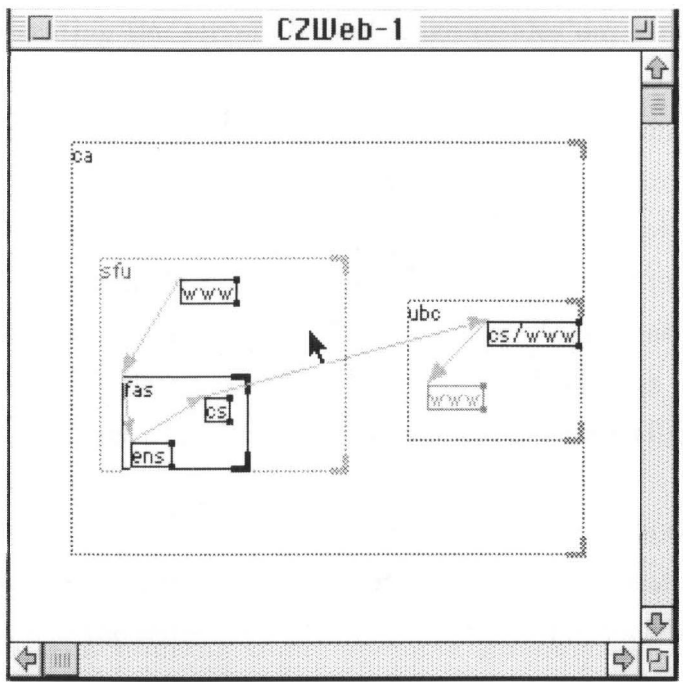

Figure 3.3: A Typical Overview of URL-tree Structure

The URL-tree structure approach, on the other hand, uses only one type of node (See Figure 3.3). Its graphical representation is similar to that of page nodes in the cluster-page structure approach. All of the nodes are resizable and have two states, closed and opened. An opened node displays its name and its children if any are inside its rectangular area. A closed node is displayed as a folder icon with its name underneath and is not resizable.

3.5 Dynamic Updating of the Overview Diagram

As the user navigates through the Web with Netscape, the overview diagram is kept updated dynamically. New nodes are added to the view as new pages are visited. Existing nodes can be removed and their relationships can be modified according to request of the user.

With the cluster-page structure, when the user visits a new Web page, a new page node is added to the overview diagram to represent that page. If there is no

cluster node representing the server the web page resides on, a new cluster node is created before the new page node is created. A link is created pointing from the node representing the previous web page to the new node. If the user returns to a visited web page, no new node is created, but the node associated to that web page is highlighted and a new link may be created if one does not exist.

The user can delete a node manually, if he is not interested in that node. When a page node is deleted, all of the arrows linking to or from that node are deleted. When a cluster node is deleted, all of its children are deleted as well. Other nodes will not affected. The user can also re-construct the automatically created structure by dragging and dropping.

One problem which occurs with this kind of dynamic system is that a small change can cause the whole view to be re-constructed which may affect the visualization because the user may lose the sense of structural continuity. The location and topological relationships among nodes serve as cues for the user to recognize visited pages and should be affected as little as possible when the user adds or deletes nodes. We made some efforts to achieve this goal.

First, we consider the location of the new node. I designed an algorithm called the "cowboy algorithm", which can determine the largest empty rectangular area inside a parent for a new node. The algorithm first divides the parent's rectangular area into small rectangles, according to its old children' sizes and locations, then unites adjacent unoccupied small rectangles to form larger ones.

The "cowboy algorithm" works well. When a new node is added in, it never overlaps with exist nodes. But it is not efficient. When the number of nodes grows, the speed declines. The other reason we do not use it any more is that the location it finds may be far away from the current highlighted node. The highlighted node represents the web page displayed on Netscape's window. When the user navigates from this page to the new page, there is some relationship between these two pages so that we hope they stay close.

The current algorithm we are using is much simpler. It works in the following way: if the new node and the previous highlighted node that represents the page displayed on Netscape have the same parent node, the new node is placed on the right side of

the previous highlighted node. If the new node and the previous highlighted node have different parents, the new node will be placed somewhere near the center of its parent.

With this algorithm, the new node may overlap with other nodes when it is added in. A "spring layout algorithm" is used to separate them later.

In the spring layout algorithm, nodes are linked with invisible *springs*, which are created based on the arrows between nodes. The algorithm to create springs is discussed in the next chapter.

Nodes are forced to move by forces created by springs. When an equilibrium is achieved, all of the nodes with springs connected to each other will stay together to form a group and nodes without springs among them will stay relatively far apart. Those patterns will help the user to recognize nodes. There should be no overlapping among nodes if enough space is available. If the user moves a node within its parent cluster node, the current equilibrium is broken. When a new equilibrium is achieved, the map may show a different pattern. If the user modifies the structure of the nodes, the structure of the springs will be modified accordingly.

After deciding the location of a new node, the next thing we need to consider is how to add it to the overview. Simply displaying the node in its location with its expected size is easy and fast. The problem is that it is too fast for the user to notice that a new node has been added in. We therefore choose to use a slower method which works as follows. When a new node is added in, its initial size is very small (about 5x5 pixels) and it grows to its default size with the Continuous Zoom algorithm. As the new node is small, it minimizes the possibility of overlapping with other existing nodes. Because we rely on the spring layout algorithm, we do not check whether its location is occupied or not. Another advantage is that it gives users some preparation for the appearance of a new node. When they watch it grow, users can easily notice that a new node has been added in as long as not a lot of other shifting is occurring.

3.6 Modify the Default Structure

The structure in the overview diagram automatically created by CZWeb is based on the URL structure and file system structure. Because people may have different opinions about the shape of a structure to represent the Web, CZWeb allows users to modify the automatically created structure manually by dragging and resizing. For example, CZWeb does not build any content based structuring, but lets users do this manually. Cluster nodes can be created manually to organize Web documents on some special topic in any way a user wishes.

Arrows links two page nodes when they are created. Users can remove existing arrows but they can not add new links manually. They are also not allowed to modify the direction an arrow.

3.7 Mapping Attributes of Nodes

In our *Cluster-page* structure version, cluster nodes have names automatically assigned as the server names, but the user is allowed to modify them. Normally, the names of cluster nodes are not very long and they can be displayed in the title bar for opened cluster nodes and under the folder icon for closed cluster nodes.

Page nodes have three labels: URL, Title and Name. The URLs and Titles are from the web pages. The Names are assigned by users and can be any strings, but users need not assign a name to each page node. The default name of a node is the same as its title. The user can choose to display any of the three labels.

Both the URL and Title of a web page are long strings and displaying long strings aesthetically on the screen is a challenging task.

CZWeb represent page nodes with resizable rectangles and display the strings inside, with long strings divided into a maximum of three lines. The first line holds characters copied from the beginning of the string until the total width of those characters is larger that 40 pixels. The second line starts from the next word from the original string and holds as many characters as possible until the total width exceed 40 pixels, and so is the third line. For example, the title "School of Computing Science"

may be split into such three lines: "School o", "Computin" and "Science". A page node has a maximum and minimum size. A page node with maximum size, which is about 30x40 pixels, can display three lines of characters in full. A page node with minimum size, which is about 10x20 pixels, can only display the first four characters of the first line.

The characters truncated in this way seem to be readable and make sense. Further, low aspect ratio rectangles are easier to manage with the Continuous Zoom algorithm and spring layout algorithm than long and narrow boxes.

In addition to the static text displayed, a pop up window is used to display the title of a page node and the name of a cluster node. If the mouse is moved on a node and stays there for a moment, a window with the name or title will be pop up. The pop up window disappears if the mouse is moved.

3.8 Other User Interface Considerations

CZWeb is implemented on the Macintosh platform. Its user interface is designed to have the "look and feel" of typical Macintosh applications. The design also follows Marcus' principles [Marc90] to achieve effective visual communication. For example, menus and dialogue boxes are designed to take into account both the content to be displayed and the screen resolution. Only two fonts and two sizes are used for all the text displayed.

Similar control handles and manipulation methods are provided for both page nodes and cluster nodes. The 'resize' handle is inconsistent with that of Macintosh window, however, because the nodes may be very small and do not have enough space for a constant size square handle. We therefore chose to use Open Window's resize handle, which takes less space. This also follows Marcus's fourth aspect of consistency: "when not to be consistent."

'Resize' handle and 'zoom' handle have different functions and are distinguished via different cursor shapes when the mouse is moved onto them. When the mouse is pressed inside a node, the cursor is changed to show the affordance of movement.

Color is very difficult to use. Taylor [Tayl86] comments on color in this way: "Color

can be a powerful tool to improve the usefulness of an information display in a wide variety of areas if color is used properly. Conversely, the inappropriate use of color can seriously reduce the functionality of a display system." Marcus's recommendation is: "Use appropriate highlighting and deemphase techniques to convey meaningful semantic distinctions." Based on such guidelines, only four colors are used: black, red, blue, and green. Black is used for normal drawing and red and blue for highlighting. Red is used for the page shown on Netscape currently. It is the most important node and should be outstanding. Blue represents 'selected'. The user can select nodes and edges to manipulate them, such as modify their attributes or delete them. More than one item can be selected at one time. Normal edges are displayed as green. Marcus emphases that "The basic idea is to use color to enhance black-and-white information, that is, design the display to work well first in black-and-white." Our current version is based on black-and-white and no background color has been added yet.

In earlier versions, sound was used to indicate that no more space is available for zooming. However feedback showed that it was annoying to the user, so that it was removed from later versions.

We discussed CZWeb's design goals and features in the chapter. The next chapter will discuss the important algorithms used in CZWeb in detail.

Chapter 4

CZWeb - Internal Operation and Algorithms

This chapter discusses detail of algorithms designed to implement CZWeb. First I shall discuss the scheme used to communicate with Netscape Navigator, then describe our algorithm to build the URL-tree, a structure used to organize visited web pages. The structure is displayed on screen with the Continuous Zoom algorithm, which is extended to meet CZWeb's special requirements. A simple spring layout algorithm is discussed last which is used to make minor adjustments to the location of nodes.

4.1 Communication with Netscape Navigator

CZWeb is implemented on Macintosh platform with Metrowork's CodeWarrior. It functions as a "companion" program to Netscape, with communication between Netscape Navigator and CZWeb occurring through Apple Events. When it starts, CZWeb checks whether Netscape Navigator is running. If Netscape Navigator is not running, CZWeb will ask the user to launch it. Netscape Navigator provides an API [Nets94] (application programming interface), which allows other applications to communicate with it and control its behavior. CZWeb registers itself to Netscape Navigator as an "echo" application, which asks Netscape Navigator to send the URL with "URL echo event" whenever it downloads a web document. CZWeb can also send Netscape

Navigator *AppleEvents* and ask it to download and display a web page.

4.2 Building a URL-tree

CZWeb uses a hierarchical structure, called a URL-tree, to organize nodes. The URL tree is built based on the URLs of the web pages the nodes represent. In order to build the URL-tree, each URL is split into an ordered list of items based on its structure. Each of these items is inserted as a unit into the *URL-tree*, which starts as an empty tree.

Let us use the following set of web pages as an example to show how the URL-tree is built. It is a subset of web pages in SFU's Web site. A Web site is defined as a group of related web documents. Those documents may be located in one or more than one web server.

1. http://www.sfu.ca/
2. http://www.sfu.ca/communication/
3. http://fas.sfu.ca/
4. http://fas.sfu.ca/ensc/
5. http://fas.sfu.ca/ensc/people/
6. http://fas.sfu.ca/cs/

For these six pages, the URL-tree built with this algorithm looks like Figure 4.1.

The URL-tree shown in Figure 4.1 is a complete tree to represent the structure of the above set of pages. We will have a tree like this after page 2, 5 and 6 having been visited.

But users can only visit pages one by one and may not want to visit all of the pages. It is not necessary to build the complete URL-tree after users visit only one page. I designed an algorithm, which will build the URL-tree incrementally as users

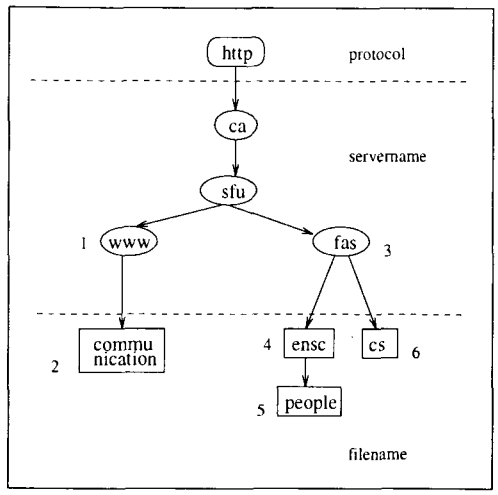

Figure 4.1: The Complete URL Tree

proceed. Different URL-trees may be formed to represent the same Web site according to different visiting sequences.

If the user visits four pages in the above set of pages in the sequence 4, 5, 6, and 2, the URL-tree is built in the following way. At the beginning, the URL-tree is empty. When page 4 is visited, its URL $http://fas.sfu.ca/ensc/$ is split into the ordered list: *http, ca, sfu, fas, ensc.* All of the items in this list are inserted into the URL-tree, which then looks like diagram 1 in Figure 4.2(a). When page 5 is visited, the list we get from its URL is *http, ca, sfu, fas, ensc, people.* When these six items are inserted into the URL-tree, only the last item, *people*, is actually added to the URL-tree, because the first five items are already there. After this, the URL-tree looks like diagram 2 in Figure 4.2(a). When the user visits more pages, the URL-tree will grow in this way. See diagram 3 and 4 in Figure 4.2(a) for how it grows after the user visits page 6 and page 2.

Figure 4.2(a) is the internal representation of the URL-tree. It is possible to represent each node in the URL-tree with a graphical node on the screen. The problem is that we may get many unnecessary nodes on the screen. For example, URL **http : //fas.sfu.ca/ensc/people/** could be split into a list with six items. If CZWeb displays six nodes to represent this single page, the user might get confused. The URL-based structuring algorithm creates nodes on the screen only when they are

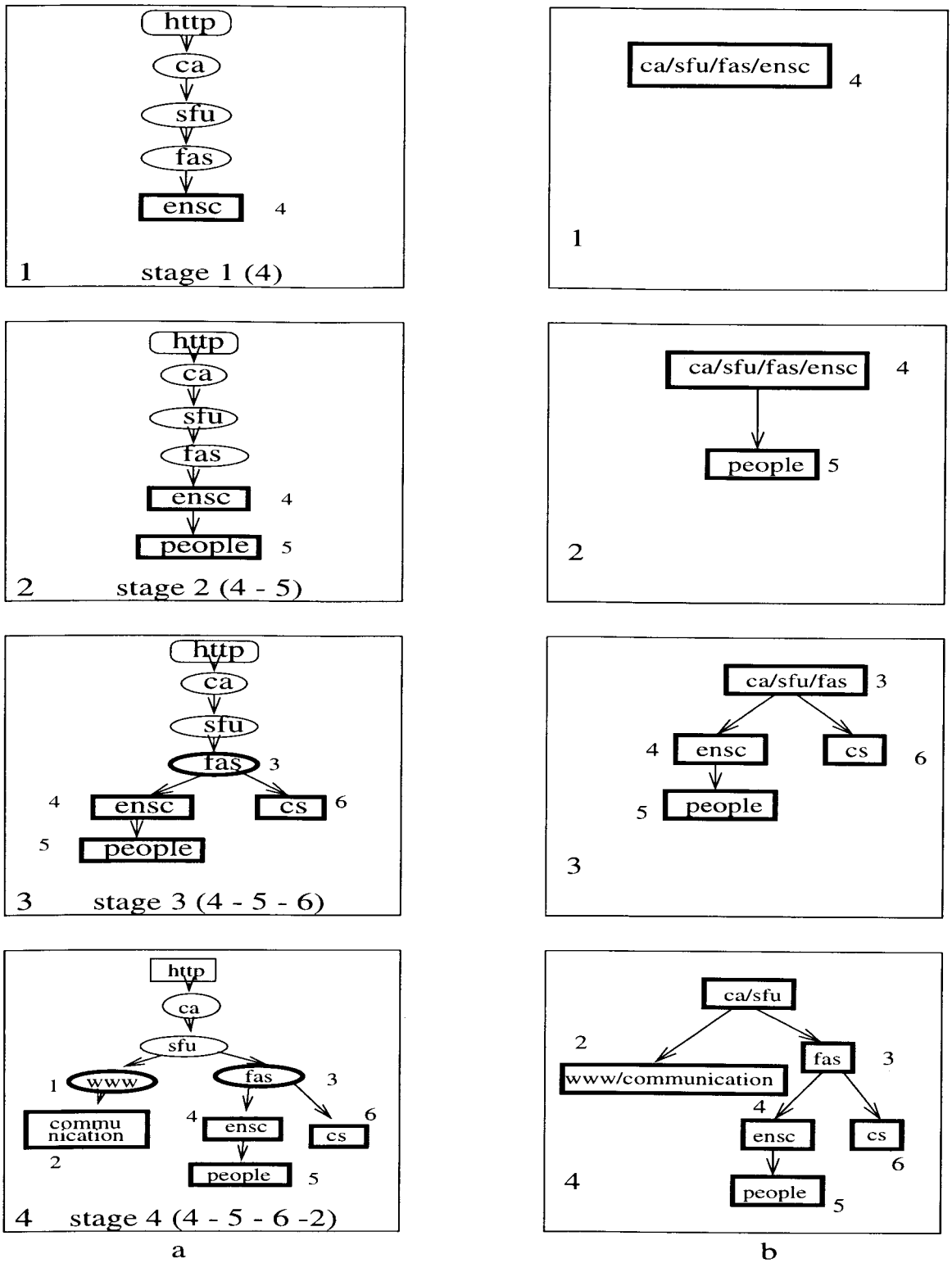

Figure 4.2: Comparison of the Internal URL-Tree and the External URL-tree

needed. Nodes may be used to represent a web page or a server. They may also be used as a container to organize other nodes. In order to do this, the algorithm creates another tree to represent nodes displayed on screen which is called the *external URL-tree*, see Figure 4.2(b). I shall refer to the tree in Figure 4.2(a) as *internal URL-tree*.

When items are inserted into the internal URL-tree, the last item in the list is marked (it is outlined with a thick line in Figure 4.2(a)), indicating that we need to add a new node in the external URL-tree to represent this new page. Note that adding a new node may result in its parent also being marked.

The new node is added in the following way. Starting from the marked node in the internal URL-tree, we search backward up the tree until an ancestor is met, which satisfies one of the following three conditions:

1. it is marked
2. it is the root
3. it is not marked but has two children (if a node has more than two child nodes, it should have been marked.)

For case 1 and 2, a new node is added to the external URL-tree as a child node of the node in the external tree which corresponds to that ancestor found in the internal URL-tree. Then we link this new node to the marked node in the internal URL-tree. The name of the newly created node in the external URL-tree is formed by combining all of the items in the internal URL-tree we passed by during the backward searching.

For case 3, since there is currently no corresponding node in the external URL-tree for the ancestor found, we need to create one (let us refer it to the *new parent node*) in the external URL-tree before we can add any children to it. It is a recursive process and the way to create the *new parent node* is the same as we just described.

The new parent node is inserted into middle of the hierarchy, rather than the leaf level. This kind of structure modification will affect nodes located below the new parent node's parent in the hierarchy. All those nodes are pushed down one level in the hierarchy. But we need only modify the location of one node.

When the *new parent node* is inserted, *its* parent node has one child node (referred to as a *sibling node*). The *sibling node* is the only one we need to modify. Its parent should be changed from the *new parent node*'s parent node to the *new parent node* and its name should be modified to fit into the new structure.

For example, at the beginning, both the internal and external URL-trees are empty. When the user visits page 4, five items are inserted into the internal URL-tree. The last one **ensc** is marked. Then we search backward up the internal URL-tree, the root is met. A new node is added to the external URL-tree as a child to the top level window, which is the root of the external URL-tree and corresponds to the root of the internal URL-tree. See diagram 1 in Figure 4.2(b). This new node corresponds to the marked node, *ensc*, in the internal URL-tree.

The name of this new node, **ca/sfu/fas/ensc**, is formed by combining all of the nodes passed by in the internal URL-tree. The nodes passed by are *ensc, fas, sfu*, and *ca*. Because *http* is the default protocol, it is not added to the name.

When the user then visits page 5, its URL is split into six items. The last one *people* is marked and inserted into the internal URL-tree. When we search back from *people*, *ensc* is met which is marked. Then a new node, **people**, is added to the external URL-tree as a child node to node **ca/sfu/fas/ensc**, which corresponds to node *ensc* in the internal URL-tree. See diagram 2 in Figure 4.2(b).

When the user then visits page 6, item *cs* is marked and inserted into the internal URL-tree. Node *fas* is met by searching back from *cs*. *Fas* is not marked yet but it has two children *ensc* and *cs*. We have to create a node in the external URL-tree to represent it before we can handle the *cs* node. We start from *fas* and search back up the internal URL-tree and this time the root is met. A new node is added as a child node to the root of the external URL-tree to correspond to node *fas* in the internal URL-tree; its name, **ca/sfu/fas**, is formed by combining the names of the node we passed by, *fas, sfu, and ca*. Then we modify the root's former child node, **ca/sfu/fas/ensc**, to be a child node to the new created node **ca/sfu/fas** and modify its name to **ensc**. Now we can add node **cs** as a child to node **ca/sfu/fas**.

Note that node **ca/sfu/fas** is created for structuring and it does not associate to any web page at the current stage.

The situation for visiting page 2 is similar to that of page 6. See Figure 4.2(b) for each of the steps in creating the external URL-tree.

The result of this stage of processing is a hierarchical tree representing, at any stage, the nodes visited so far. The tree also indicates a name for each of the nodes to be displayed. In the following sections, we describe how these nodes are laid out in the display window.

4.3 Displaying the URL-Tree

The previous section described building a structure to organize nodes. In this section we describe an algorithm we developed to show this structure visually. The algorithm is an extension of the Continuous Zoom algorithm[Dill95].

4.3.1 Background

As we discussed section 2.4, there are many techniques to display hierarchical structures. The main challenging task is to solve the cluttering problem. The screen size is limited so that when the number of nodes increase, it is impossible to display all of them visibly on screen.

The Continuous Zoom algorithm indicates hierarchy by a geometric containment relationship. It is a *detail-in-context* algorithm, which displays a particular portion of a hierarchical structure at a greater level of detail while still displaying all or much of the context (the overall structure). But it has some shortcomings that affect its effectiveness in this application:

- It can control nodes growing and shrinking, but cannot explicitly control the exact size of each node
- When any node grows or shrinks, all of the nodes in the space change size

CZWeb requires the algorithm to have the ability to control the exact size of any node, and to let some nodes keep constant size when other nodes are zoomed in or

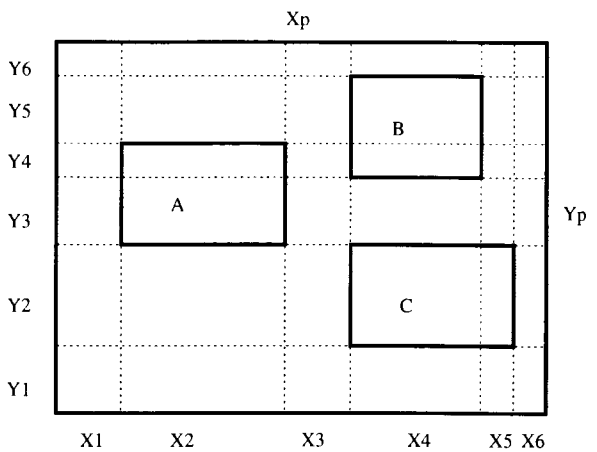

Figure 4.3: The Original Continuous Zoom Algorithm

out. In the rest of this section, we first review the original CZ algorithm, then describe our extension to it which solves the shortcomings mentioned.

4.3.2 Review of the Original CZ Algorithm

The original CZ algorithm [Dill95] works independently in X and Y axes. It first projects all node boundaries, or edges, onto the X and Y axes. *Intervals* are the spaces between the "grid lines" created by these projected edges (the X_i and Y_i in Figure 4.3). A scale factor is assigned to each node to specify its size change. For example, to expand node A in Figure 4.3, it could be assigned a scale factor S_a, greater than 1, and scale factors of unity would be given to nodes B and C. Then the algorithm computes a scale factor for each interval. Each interval is either a projection of one (or more) nodes, or is an 'inter-node" or "gap" interval. The scale factor of a projection interval is the maximum scale factor of all the nodes that project onto it. There are several options to set the scale factors for "gap" intervals. The easiest way is to set them to 1. The total amount of space requested by the cluster (in the X direction) is X_{req}

$$X_{req} = \sum_i X_i s_i$$

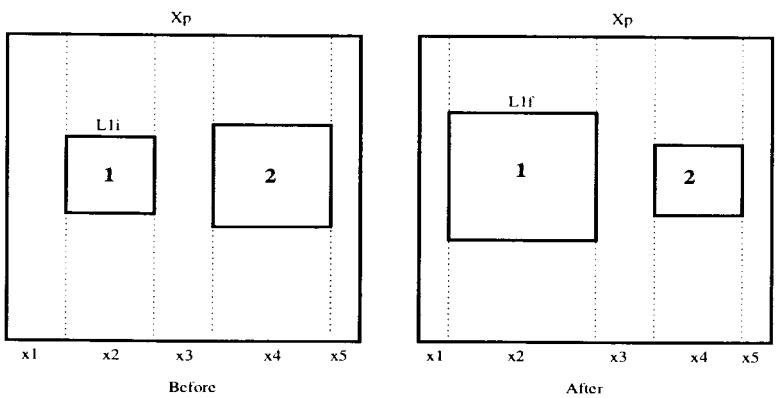

Figure 4.4: Zoom node 1 to the exact size and node 2 is resizable

Suppose the cluster cannot expand. The resulting scale factors for node A, B and C and gap intervals should be their initial scale factors times S_p,

$$S_p = \frac{X_p}{X_{req}}$$

where X_p is the length of the cluster node in X.

With this algorithm, we can expand and shrink a node, but we cannot control the exact size of each node. I have extended this algorithm to allow the user to zoom a node in or out to an exact size (the size and location of other nodes will be adjusted appropriately to efficiently use the screen space). For example, to close a node, it is de-magnified (zoomed out) to its default closed size. To open a node, it is zoomed in to the default opened size. In addition, the new algorithm allows some of the nodes (closed nodes displayed as folder icons) to keep a constant size as other nodes are zoomed in or out.

4.3.3 A Simple Case: Exact Size Zoom

We start with a simple case: We want to resize node 1 from size L_{1i} to size L_{1f}, assuming all of the nodes are resizable. See Figure 4.4. If we just assign node 1 the scale factor $s = L_{1f}/L_{1i}$, it would not work because node 1 will be scaled by a factor s', which is smaller than s:

$$s' = s\frac{X_p}{X_{req}}$$

In order to let node 1 scale by exactly s, we should assign its initial scale factor to s_i, where s_i satisfies:

$$s_i\frac{X_p}{X_{req}} = s \tag{4.1}$$

where

$$s = \frac{L_{1f}}{L_{1i}} \tag{4.2}$$

In Figure 4.4, scale factor of interval x_2 is s_{x_2} which is equal to s_i and all of the other intervals' scale factors are 1. In this case, we do not consider overlapping nodes, as this will affect computing the scale factors of intervals. We shall take care of this in later examples.

By definition, we have X_p and X_{req}

$$X_p = \sum_i x_i = x_1 + L_{1i} + x_3 + x_4 + x_5 \tag{4.3}$$

$$X_{req} = \sum_i x_i s_{x_i} = x_1 + L_{1i}s_i + x_3 + x_4 + x_5 \tag{4.4}$$

By combining Equation 4.1 and Equation 4.2, we can get:

$$L_{1i}s_i = \frac{X_{req}L_{1f}}{X_p} \tag{4.5}$$

By solving Equation 4.3, we have:

$$x_1 + x_3 + x_4 + x_5 = X_p - L_{1i} \tag{4.6}$$

Replacing Equation 4.5 and Equation 4.5 into Equation 4.4, we can get:

$$X_{req} = X_p - L_{1i} + \frac{X_{req}L_{1f}}{X_p} = \frac{X_p(X_p - L_{1i})}{X_p - L_{1f}} \tag{4.7}$$

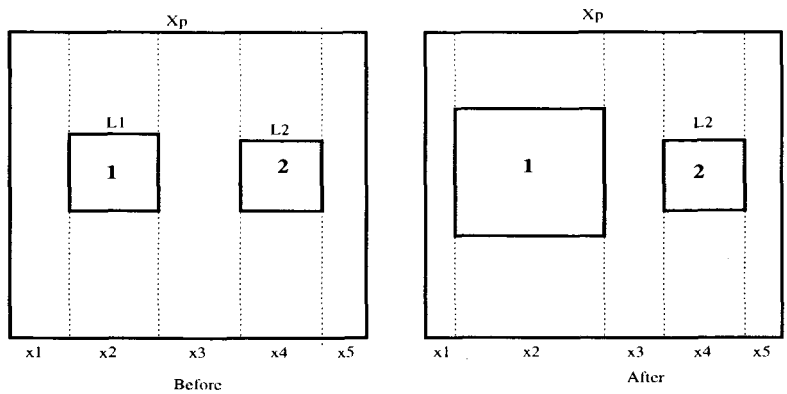

Figure 4.5: Single Size Constant

Replacing Equation 4.7 into Equation 4.5, we can get s_i at last:

$$s_i = \frac{X_{req} L_{1f}}{X_p L_{1i}} = (\frac{L_{1f}}{L_{1i}})(\frac{X_p - L_{1i}}{X_p - L_{1f}}) \tag{4.8}$$

4.3.4 Case 2: Single Size Constant

Now, let us consider our second case, we want to zoom node 1 by scale factor s_{1_i}. We do not care about its final exact size, but node 2 should remain a constant size. See Figure 4.5. We assign scale factor s_{1_i} to node 1 and s_{2_i} to node 2. The initial size in the X direction of node 1 and node 2 are L_1 and L_2 respectively. S_{2_i} is unknown and we are going to solve it.

Now the total amount of space requested should be:

$$X_{req} = L_1 s_{1_i} + L_2 s_{2_i} + L_g 1 \tag{4.9}$$

where L_g is the length of the "gap" intervals and

$$L_g = X_p - L_1 - L_2 \tag{4.10}$$

In order to let node 2 keep its size, we should have $s_{2_f} = 1$, but

$$s_{2_f} = s_{2_i} \frac{X_p}{X_{req}} = 1 \tag{4.11}$$

We solve Equation 4.9, 4.10 and 4.11 and get s_{2_i}:

$$s_{2_i} = \frac{L_1 s_{1_i} + X_p - L_1 - L_2}{X_p - L_2}$$

4.3.5 Case 3: Exact Size Zoom With Size Constant

This algorithm combines the above two. Node 1's initial size is L_{1_i} and it should be resized to L_{1_f}, but node 2 should keep its constant size L_2. In order to calculate initial scale factors s_{1_i} and s_{2_i}, we can write down the similar equations:

$$s_{1_f} = \frac{L_{1_f}}{L_{1_i}}$$

$$s_{2_f} = 1$$

$$X_{req} = L_{1_i} s_{1_i} + L_2 s_{2_i} + (X_p - L_{1_i} - L_2)1$$

$$s_{1_i} \frac{X_p}{X_{req}} = s_{1_f}$$

$$s_{2_i} \frac{X_p}{X_{req}} = s_{2_f}$$

Solving the above set of equations, we get:

$$s_{1_i} = \frac{\frac{L_2 s_{1_f}(X_p - L_{1_i} - L_2)}{L_2 - X_p} - (X_p - L_{1_i} - L_2)s_{1_f}}{L_{1_f} s_{1_f} - X_p - \frac{L_2 L_{1_f}}{L_2 - X_p}} \tag{4.12}$$

$$s_{2_i} = \frac{L_{1_i} s_{1_i} + (X_p - L_{1_i} - L_2)}{X_p - L_2} \tag{4.13}$$

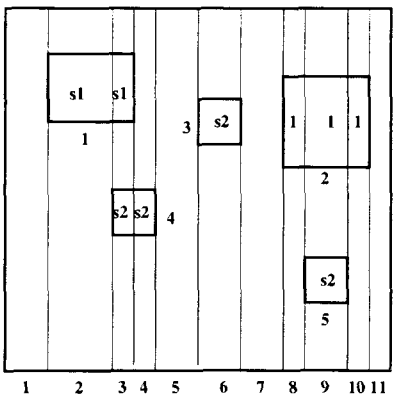

Figure 4.6: Complex Case

4.3.6 Complex Cases

Now let us consider a more complex case, with overlapping nodes (see Figure 4.6). It is the real case that we need to handle in CZWeb. We want to zoom node 1 from size L_{1_i} to size L_{1_f}, we do not care about node 2's size, but node 3, 4 and 5 should remain constant size. Variables we want to solve are still s_{1_i} and s_{2_i}. But it is difficult to calculate the X_{req} and Y_{req} now.

Suppose scale factor s_{1_i} for node 1 is greater than 1. Node 2 and the "gap" intervals shrink, with their initial scale factors of 1. For the three remaining nodes, s_{2_i} is assigned to their scale factors and is required to satisfy

$$1 < s_{2_i} < s_{1_i} \tag{4.14}$$

This is because if $s_{2_i} \leq 1$, they will shrink and if $s_{2_i} \geq s_{1_i}$, they will grow. This also ensures the scale factor for interval 3 is the same as the scale factor of node 1, and the scale factor for interval 9 is the same as the scale factor of node 5.

We make an initial estimate of s_{2_i} as:

$$s_{2_i} = \frac{1 + s_{1_f}}{2}$$

where $s_{1_f} = L_{1_f}/L_{1_i}$.

Initially, the scale factors for all the intervals from 1 to 11 are set to 1. The scale factor for an interval must be the maximum of the scale factors of the nodes projecting

onto that interval. In this case, the scale factors for the intervals should be:

interval	1	2	3	4	5	6	7	8	9	10	11
scalefactor	1	s_{1_i}	s_{1_i}	s_{2_i}	1	s_{2_i}	1	1	s_{2_i}	1	1

Let L_k be the summation of the sizes of intervals 4, 6 and 9, i.e. where $s = s_{2_i}$. Now we can replace L_k with L_2 in Equation 4.12 and 4.13 and solve for s_{1_i} and s_{2_i}.

After solving for s_{1_i} and s_{2_i}, we assign s_{1_i} to node 1 and assign s_{2_i} to node 3, 4 and 5 and use the original Continuous Zoom algorithm to calculate the size of each node. The final result will satisfy our requirement.

4.3.7 Some Implementation Considerations

The algorithm we described in the previous section considers the X direction only. Although those equations can be applied to the Y direction, aspect ratios of nodes can not be kept as constant if we apply the algorithm on both directions at the same time.

In our implementation, the scale factors of gaps are always assigned to 1. This has two disadvantages. First, the size of gaps decrease all the time no matter which node is zoomed. When the gap between two nodes is less than one pixel, those two nodes overlap. Then there is no gap any more and the amount of overlap increases or decreases with these two nodes. Second, a node may move out of its parent's boundary when the gap between the node and the boundary disappears. This problem is easier to solve than the former one, because we can easily control nodes' size and location and do not allow them moving out of their parents' boundaries. But this solution will cause another chance of overlapping if more than one child node moves close to the same boundary.

In the IGI project's implementation, the scale factor of a gap is assigned to the largest scale factors of all adjacent nodes. In this way, gaps do not decrease all the time. If one of its adjacent nodes increases, it grows too. But this algorithm has a limitation. When a node is the only child node of its parent, it is impossible to change its size with this algorithm. Because in this case, all intervals have the same scale factor. The result of Continuous Zoom will not change anything. This special

situation did not appear in the IGI project, but it does happen very often in CZWeb (when a web server is accessed at the first time, there is only one page node inside the cluster node representing that web server) so that we cannot use this algorithm.

One possible solution is checking the number of children when considering the scale factor for gaps. If there is only one child, assign unit to scale factor of gaps. Otherwise use the IGI project's algorithm.

Another problem in CZWeb's implementation is that after we zoom in a node, we cannot go back exactly to its original location and size by zooming it out. This is because of a number of unnecessary round-offs from float to integer of nodes' sizes and locations in the implementation. In order to provide a sense of visual continuity among states, zooming a node from one size to another size is divided into a sequence of zooms. The problem in our current implementation is that those zooms in the sequence are *independent* of each other. Although floats are used inside the Continuous Zoom algorithm, the size and location of nodes are saved as integers. Each time the Continuous Zoom algorithm takes integer locations and sizes and calculates new locations and sizes in float. But the floats are rounded off to integers and saved. Next time the Continuous Zoom algorithm has to use integers again. After many zooms, those round-offs will make a large difference.

The spring layout algorithm will not affect the Continuous Zoom algorithm, because the spring layout algorithm is running at a low priority.

A better way to implement the Continuous Zoom algorithm should keep the locations and sizes as floats all the time. After each zoom step, round them off to integers and display the nodes at their new locations, but save floats rather than integers for the next zoom. I believe this will improve the performance.

4.4 A Simple Spring Layout Algorithm

The (extended) Continuous Zoom algorithm controls the size and location of every node. But sometimes nodes overlap each other, e.g. a new node may overlap existing nodes when it is added in. CZWeb uses an additional algorithm, a simple spring layout algorithm, to adjust locations and movements of nodes.

Formal spring-model algorithms for graph layout are discussed in [Kamp95]. Our simple spring layout algorithm is based on a Java applet demonstration program[Java96]. This applet consists of a 2D window with about a dozen nodes linked with edges. In this applet, nodes are displayed as constant size rectangles (about 10x40 pixels), but are actually treated as points. Nodes have names displayed inside their rectangles. Users can move nodes by dragging and dropping.

We modified this algorithm to allow for variable node size and to support a hierarchical structure. It runs at a low priority and arranges nodes in a 2D display area so as to minimize overlapping and also to group nodes together.

The original algorithm is based on the edges between nodes. In order to deal with the hierarchical structure of our nodes, a new data type, *spring*, is created and it is the basis of our spring layout algorithm.

Springs are created based on links between nodes but there is not a one-to-one relationship. They are internal to the algorithm and are not visible to the user. Springs are organized into a hierarchy too; each cluster node may contain its own set of springs and the spring layout algorithm based on this set of springs only affects movement of this cluster node's children.

For example, in Figure 4.7, there are three cluster nodes, C_1, C_2 and C_3, and seven page nodes, P_{0_a}, P_{1_a}, P_{1_b}, P_{2_a}, P_{2_b}, P_{3_a}, P_{3_b}.

Four spring layout algorithms are running, one for each cluster and one for the top level window. The algorithm on the top level window manages the locations and movements of node C_1, C_2 and P_{0_a}. The algorithm in cluster C_1 manages page nodes P_{1_a}, P_{1_b}, and so on.

The thin arrows in Figure 4.7 represent the edges between page nodes. Springs are drawn as thick lines. They are created based on the following rules.

1. For each arrow, find the common ancestor of the two nodes linked by the arrow. Then add a spring to the spring set of that ancestor. This spring links the two children of the common ancestor. Those two children nodes may be the two nodes linked by the arrow or their ancestors. (I use two stacks to implement this.) For example, arrow L_1 links page nodes P_{1_a} and P_{2_a}. The common

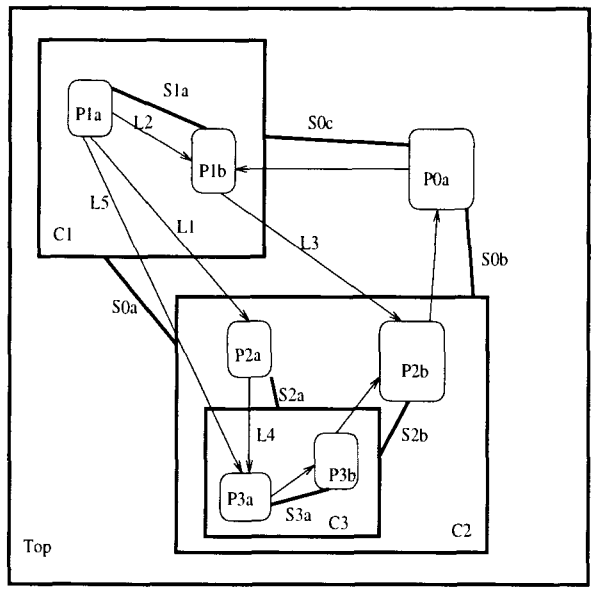

Figure 4.7: Spring Layout Algorithm

ancestor for these two page nodes is the top level window. Spring S_{0_a} is created for the top level window to link nodes C_1 and C_2, which are two children of the top level window.

2. No matter how many arrows exist, there is at most one spring between any pair of nodes. For example, when arrow L_5 is processed, no new spring is created because spring S_{0_a} exists, but the *duplication* attribute of spring S_{0_a} increases by one.

Each spring has a *preferred length*, which is determined by a constant and the size of the two nodes it connects. If the distance between the centers of the two nodes linked by a spring is longer then the spring's preferred length, there is an attractive force pulling these two nodes together along the direction of the spring. Otherwise, there is a repulsive force that pushes these two nodes away. The larger the difference between the distance and the preferred length, the larger the force.

There are repulsive forces among overlapped nodes too. The direction of these forces is along the line of the centers of the two overlapped nodes.

The movement of a node is determined by the summation of all of the forces applied

on it. When an equilibrium is achieved, all of the nodes with springs connected to each other will form a group with some special pattern and different groups can be moved independently. There should be no overlapping among nodes if enough space is available.

If the user moves a node within its parent cluster node, the constant part of the preferred lengths of all springs attached to the moved node will be modified. When a new equilibrium is achieved, the map may show a different pattern. If the user modifies the structure of the nodes, the structure of the springs will be modified accordingly.

The constant part of the preferred length is a different value for each spring, it depends on the *duplication* attribute of that spring. The larger the duplication, the smaller the constant. In this way, two clusters with more arrows between their children will stay closer than those with fewer arrows.

One problem with this spring layout algorithm is oscillation – some nodes move forth and back and cannot settle down. There are many forces applied on a node at one location. If we move the node to a new location, a new set of forces will apply to it because it may overlap with different nodes. If the resultant force is in the opposite direction, oscillation may occur. In order to let nodes move smoothly, nodes' movements are limited with the maximum length one step can take. For example, a node in the current location is required to move eight pixels left, but the maximum length each step can move is five pixels, so that it moves left five pixels. In its new location, the node is required to move right ten pixels, but it can only move five pixels at one step so that it moves right five pixels which moves it back to its original location – oscillation happens. We do not have a good solution to this problem. Currently we count the oscillations and if a node has moved back and forth five times, we stop the spring layout algorithm at that level. According to our observation, five steps are enough for nodes to settle down if there are no oscillation situations. For later improvement, we should consider replace this simple algorithm with a formal spring layout algorithm.

Chapter 5

CZWeb Evaluation

Evaluation is a very import part of software development and we have spent much effort on it. CZWeb (the cluster-page structure version) has been evaluated three times by four groups. This chapter summarizes the processes and results of those evaluations.

5.1 First Evaluation

The first evaluation was done by a group formed by my classmates and myself as a term project for a graduate HCI course offered by Dr. Tom Calvert (Simon Fraser University, CMPT-882) and Dr. Kelly Booth (University of British Columbia, CPSC-533) in the 96-1 semester. The result was discussed in project report [Bani96].

The goal of this study was comparing users' mental model about the Web with CZWeb's design model and testing CZWeb's functionality, usability and acceptability. We also tried to identify problems encountered by potential end-users and to determine what proportion of the system's functionality is utilized.

We categorized Web users into two types: *casual users* and *expert users*. Casual users are new to the web and have more chance to be confused and get lost in the Web. Expert users are adept at using web browsers and understand navigation techniques, but need an efficient organizational tool to aid them in their navigation. Because it is the intention of CZWeb to aid all users in their exploration of the Web, it is important

to know how users of different types feel about web browsers and CZWeb.

We designed a "treasure hunting" game to study users' browsing strategy. It had four small activities. The first one was designed for casual users to familiarize themselves with the web browser. The other three activities were questions. Each question contained a starting web page and some clues. Subjects were required to find the answer to the question by browsing the web site within ten minutes. No search engines were allowed. The URL of the index page to the game is in [Game].

We also had two interviews, one before the "treasure hunting" game and the other after the game. The first interview was to collect information about the users' background and their general opinions about web and web browsers.We interviewed five users. Two of them were casual users and the other three were expert users.

After the first interview, the five users were divided into two groups to perform the "treasure hunting" game. Group one consisted of two users, one casual user and one expert user. They were required to use both Netscape and CZWeb to navigate. The screen was divided into two parts horizontally. The Netscape window located on the left and CZWeb window on the right. Users were allowed to use facilities provided by either Netscape or CZWeb to access visited pages. Group two consisted of the remaining three users. They used Netscape only and were required to draw their own map on paper to represent the structure of web pages they visited.

The interview after the game for group 1 focused on the users' feedback of using CZWeb. The questionnaires consisted of ten questions. Users could select any number between 1 to 7 to represent their satisfaction about one aspect of CZWeb. The interview for group 2 focused on users' mental model about the Web. Users were asked to explain the maps they drew during the game.

Because of the small number of users and time limit, we did not do any statistical analysis of the experiment. Only a qualitative analysis was done based on our observation on users' behavior in the "treasure hunting" game and the interview.

Users thought the Web structure was similar to a big file system, although it is much more complex and unorganized than a hierarchical structure. They seemed comfortable navigating to two or three levels using Netscape's **Back/Forward** buttons. We speculate that they can remember a three-level tree temporarily. They went

to higher levels occasionally. When this happened, users in group one tended to use CZWeb to go back to a familiar territory by clicking on a node linked with many edges, which was the node they visited often and had access to other nodes. Users in group 2 tended to use Netscape's **Go** list rather than the **Back** button to go back to a visited web page.

The web sites we used for the 'treasure hunting' game were selected from the real world. Some pages contained dozen of links, which might be linked to anywhere in the world. The 'treasures' were located deeper than three levels from the starting page. The expert user in group 1 found two 'treasures' in the time limit. The casual user only found one. Two users in group 2 found one 'treasure'. The third user gave up after failing to find the first 'treasure' in 15 minutes.

Users in group 1 had not used CZWeb very much because they were not shown the functions provided by CZWeb before the experiment. Because of the time limit, they were in a rush in the game and did not have time to learn CZWeb. The casual user did not even notice that CZWeb had a menu bar. They were not satisfied with CZWeb's graphics and layout either, ("It is very hard to read the label of a node").

In group 2, only one user wrote a few words to represent web pages he visited at the beginning. Writing the title of web pages was time consuming, so that he did not keep doing this in the rest of the game. The other two did not draw anything on the paper. When asked how he managed visited web pages, one user said "I just remember them, but I do not know how."

This evaluation suggests that users think the Web is formed as a hierarchical structure and they can remember two or three levels of the structure when they are navigating through it. My URL-tree structure is based on this suggestion.

5.2 Second Evaluation

A more formal evaluation was later designed and undertaken and the results were published in [Fish97]. This experiment used 12 university students. Subjects were asked a series of questions to assess their experience of using different platforms and Internet tools, such as web browsers and electronic mail, and their perception on how

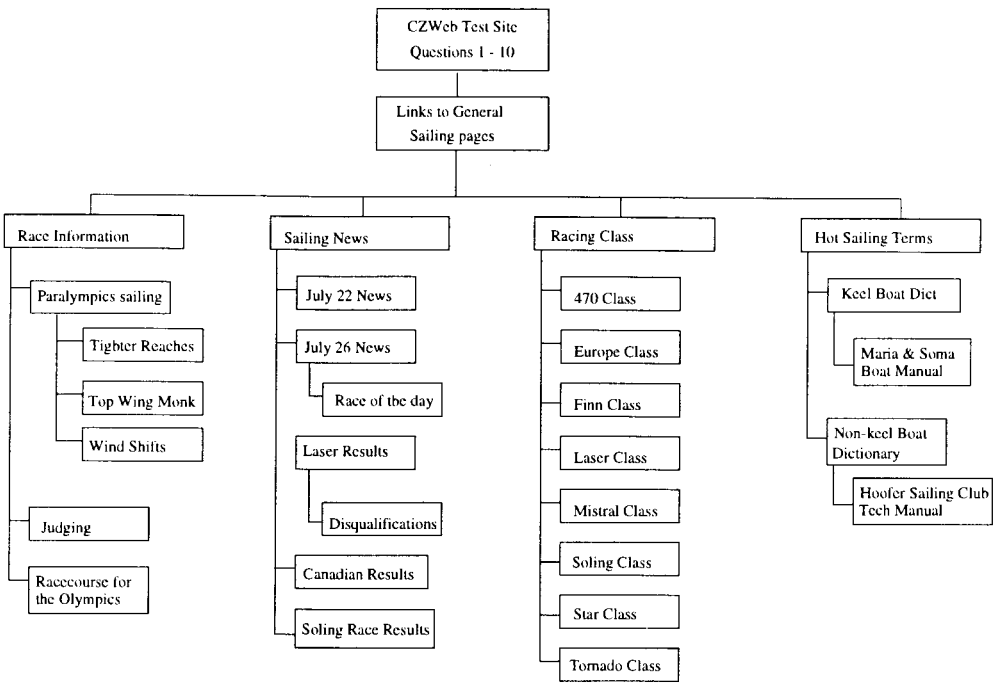

Figure 5.1: The Structure of Test Web Site

the web is used, how often and how easily they use the web.

After the questions, the experiment examined how well CZWeb helped them find information inside a web site designed for the experiment. This test web site contained information about a 1996 Olympic sailing event, including results of a race and a glossary of items. It was a very small and simple web site, which contained only 30 web pages. Those web pages were organized as a four level hierarchy. Figure 5.1 shows the structure of the nodes and links in this test web site. All of the pages were copied onto local hard disk for the experiment. A special experiment CZWeb version was used in the experiment, because the standard cluster-page structure version only created one cluster node for each web server (as mentioned in subsection 3.3.4). In this case all visited pages would be inside one cluster node representing the root directory in the local disk. The structure created by the standard cluster-page structure version is shown in Figure 5.2. The special experiment version created a cluster for each subdirectory in the second level to organize the 30 test web pages. Figure 5.3 is the structure created by the URL-tree structure version. It is similar to the structure

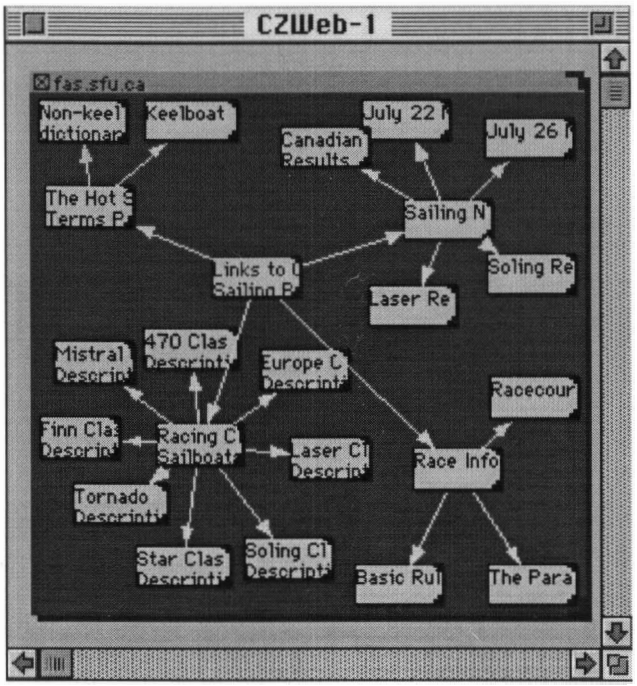

Figure 5.2: Test Site Structure Created with Standard Cluster-Page Structure Version

created by the special experiment version. But the URL-tree structure version created this structure automatically, no special adjustment is needed. The labels of all the nodes in Figure 5.2 were truncated from web pages' titles. Those in Figure 5.3 were formed from web pages' URLs. (That is why they look different).

The system was set up for users before the experiment. The screen was divided into two parts, with the Netscape window located at the top-half and CZWeb window at the bottom-half and both windows had roughly the same width. As users browsing, CZWeb created and updated a map of all the pages visited. Users were allowed to use all of the functions provided either by Netscape or CZWeb to access web pages. The progress was video taped and think-aloud protocol was used.

The subjects' tasks were to answer 12 questions, such as "Why would you do two 360 degree turns during a race?" These questions were carefully designed so that the answers sometimes required combining information from two or more web pages in the experiment web site. Subjects were shown how to answer questions by a demonstration, which answered the first two questions.

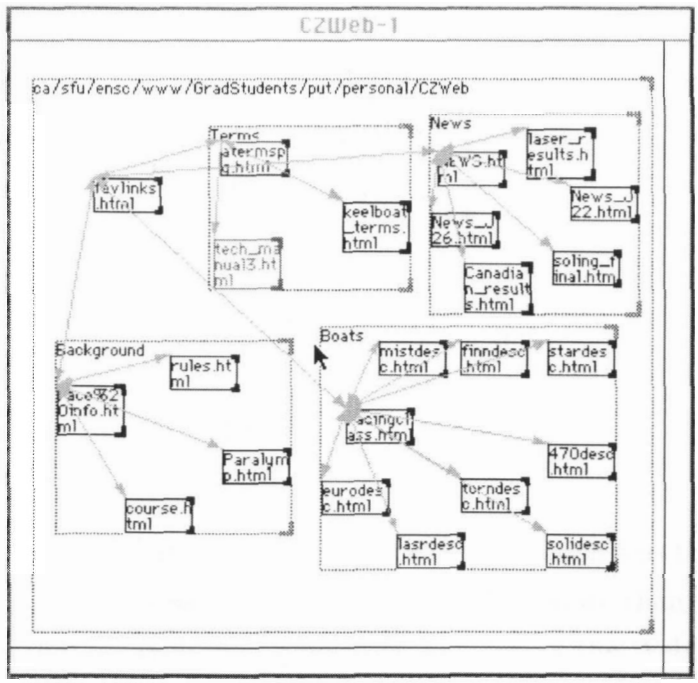

Figure 5.3: Test Site Structure Created with URL-Tree Structure Version

On finishing the task, subjects completed a post-test questionnaire, which asked them to rate various aspects of CZWeb. For example, the first question is "How do you rate CZWeb's overall ease of use"? (2 = very easy; -2 = very difficult). The data is as follows:

Group	Meaning	Count	%
-2	very difficult	0	0
-1	difficult	2	16.7
0	neutral	2	16.7
1	easy	7	58.3
2	very easy	1	8.33
total		12	100

The null hypothesis that CZWeb is neutral with regarding to overall ease of use

was tested at the significance level of 0.05. That is to test the null hypothesis: $\mu_0 = 0$ at $\alpha = 0.05$. Here t-test was used. Let x denote the answer and s denote the sample standard deviation. We compute the t-statistic as follows:

$$\bar{x} = \sum_{i=1}^{n} x_i = 0.583$$

$$s = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}} = 0.9$$

$$t = \frac{\bar{x} - \mu_0}{s/\sqrt{n}} = 2.244$$

where $n = 12$ and $df = n - 1 = 11$.

The t table gave the p_value as 0.0463. Since the p_value is less than α, we rejected the null hypothesis. Because the mean is 0.583 which is more than zero, the result is positive. It shows that CZWeb does help users navigating the Web.

On the question "Having CZWeb's picture of the web site you have visited helps you move around the Web?" the result based on t-statistic was ($t = 3.079$, 11 df, $p < 0.05$).

There are other results which show that CZWeb helps users understanding the web structure, takes less movements to recently visited sites, and helps users localizing the current position. For example, for the question "Please rank (first second third) how you would use each of the navigation tools (CZWeb, Back/forward, history) for the given application:"

- On "Helping you to understand the Web organization better", the result was positive with $p < 0.0001$.
- On "Going back more than 5 sites", the result was positive with $p < 0.0001$.
- On "Knowing where I am in the Web and avoiding going to the wrong site", the result was positive with $p < 0.0001$.

The experiment also shows that no signification effects were found for questions such as how CZWeb:

- Reflects how you think about the Web
- Makes you feel like you are traveling throughout the Web
- Makes Web organization more clear

It appears that users have a greater need to individuate web pages and isolate relationships among them to reduce disorientation and cognitive overhead. This corresponds to the survey results of [Pitk96]. As the questions become more explicitly spatial, agreement falls off. Perhaps users want to generate organizations that reflect how they think about the relationships between sites, a personal Web space, which may lack the static metrics of real spaces and maps.

5.3 Third Evaluation

This evaluation was undertaken by two groups of students as term projects in the HCI course offered by Dr. Tom Calvert, Dr. Kelly Booth, and Dr. John Dill in 97-1 semester (CMPT-873 in SFU and CPSC 533B in UBC). Results are discussed in project reports [Walk97] and [Cubr97].

These studies were based on analysis of the videotapes shot in Brian Fisher's evaluation experiment described in the previous sub-section. By observing users' action carefully and analyzing it with the help of MacSHAPA software, both groups revealed some design problems and suggested some improvements.

Three user modes had been observed. One was the Window-conservative Mode. Users tended to use the navigational method that has been used last. This mode was confused with the cause of explicit switching windows in Macintosh. Another one was the novelty mode. Novice users tended to try features of one application for a while and then try features of another application. The third was the path-memory mode. Novice users tended to memorize the path he took to get one particular page unaware of where he was or CZWeb was of help.

Each group had found some problems. The one both groups identified was the method used to open a web page. CZWeb uses double click on a page node to open

the page represent by that node, but Netscape uses single click on a link to go to the linked page. Double click to open a page is consistent with the Macintosh windows operations, but it is not consistent with the single click to activate a link. These two play the same role of getting to a particular URL page but use different operations. This led to the situation where the users double clicked to follow a link in Netscape. The second problem was that using CZWeb could only take users to those pages that the users have visited. This was not actually a problem but a suggestion. One of the groups observed that one subject in the experiment did a breadth-first search. This suggested it was important for CZWeb to be able to do a simple breadth-first search. Another group also suggested that CZWeb be able to auto-explore a web site according to a preset level.

Some other problems were trivial comparing to the above two. One was the page location. When **Back or History** was used to go to a visited page, the cursor was at the place where the user left. When back to a visited page using CZWeb, the cursor was always at the beginning of a page. Another was the problem with the label of page nodes. When there were a relatively large amount of nodes in the CZWeb window, page nodes became very tiny. It was hard to read the label. It was also hard to resize or to "zoom" them. The only way to check out a page node's label was to wait for the pop-up helps. This was awkward and slow. Another problem was with cluster nodes. Cluster nodes were used as containers to organize nodes, but users thought they might represent some web pages as well. This ends up with a scenario that a user is double clicking on the title bar of a cluster.

From the evaluation, we know that using a hierarchy to represent the web structure is consistent with users' mental models. The positive results from the evaluations shows that a representation of web's structure does help users understand the web, localize their current location, and move around the web. Facilities provided by CZWeb also helps users go back to visited web pages quickly. But, there are still some areas we need to improve, such as making users feel as if they are traveling through the web, building a clearer structure to represent the web, and improving the user interface.

All of these evaluations are valuable. The positive results tell us that we are on the

right track and encourage us to keep going. Identifying existing problems is helpful for CZWeb's further development.

Chapter 6

Future Work and Conclusion

6.1 Discussion

6.1.1 Link-based Structuring vs. URL-based Structuring

Almost all of the systems and approaches we examined in Chapter 2 build their structures based on the link relationship between nodes. When they simplify the underlying network structure of hypertext into a tree or tree-like structure, a node is considered to be a child node of another node if there is a link pointed from the latter to the former. We refer to this as "link-based structuring". The advantage of link-based structuring is that the structure it created is simple and clear. The disadvantage is that some information is lost during the structuring process.

The URL-based structuring approach, proposed in this thesis, is a new idea to represent the web structure. With this approach, all of the cross-connections are preserved to show the correct web structure to the user. The node hierarchy automatically manages the information at an understandable level and it is very easy to change the level of detail. The node hierarchy created by CZWeb is based on the structure of files stored on web sites, which are managed by webmasters and web authors. Generally, we can assume that most of the web pages are reasonably managed which means that the structure automatically created by CZWeb will not get out of control. In most of the file systems, files inside one subdirectory should have some

common attributes and parent directories should have some relationship with their subdirectories. Normally, people will not place thousands of unrelated files into one directory.

Of course, there are always different opinions on how to structure information and CZWeb allows user to modify the default created structure as described in chapter 3.

6.1.2 Hyperlinks vs. Temporal Path

When we create links in order to link nodes in our structure, two kinds of information should be taken into account: the hyperlinks embedded in the web documents which are used to link all of the web documents together and the path a user actually took to visit those web documents. Both kinds of information are important for orientation in hypertext. The hyperlinks help to understand the underlying structure of the web. The path records the historical information and is helpful for the user to figure out where he came from and how he got there.

The Hy+ displays both of them in one overview diagram and distinguishes them with colors. The whole path is recorded in Hy+ and displayed red arrows. The first travel from one node to another is displayed as a straight arrow. Later travels between these two nodes (in either direction) are displayed as broken arrows. Hyperlinks between nodes are displayed with black arrows. In the way, it seems very clear for both the hyperlinks and the path. But, the problem is that the overview diagram will be cluttered with arrows very soon.

In CZWeb, both of the hyperlinks between nodes and the path are displayed as straight arrows. There is at most one arrow between any two nodes. Duplication of hyperlinks and path are not recorded. But cluttering is still a big problem. For example, one of the subjects in the second evaluation was an experienced Netscape user and used Netscape's "Go" list very often. Almost every pair of the nodes in the view was linked together after he used it for a while. It was hard to trace the path from the cluttered view.

Our current solution is adding a time stamp on each edge in the path. The user can choose letting old edges dim out or displaying only the most recent edges. We

can control the number of edges displayed easily this way.

6.1.3 Distinguish User's Actions

With Netscape, the user can view a page by different actions:

- Clicking on a hyperlink embedded in the current page pointed to that page
- Typing in the URL of the page directly
- Selecting the destination from the "Go" list if it is visited recently
- Selecting it from the "Bookmarks" if it was saved there

It is good if we can distinguish these four user actions. WebJournal did this at the cost of modifying NCSA Mosiac.

CZWeb uses Netscape's API to communicate with Netscape. When Netscape displays a new web page, it sends an AppleEvent to CZWeb to notify the URL of the page it displays. CZWeb can also send an AppleEvent to Netscape and ask it to display a special web page. But that is all we can do. We have not been able to access user actions (mouse clicks and keyboard typing) inside Netscape's window so that we cannot distinguish the actions the user took to get a new web page.

We can get a set of URLs accessible from the current page by parsing the HTML file of the current web page. This information can be used to build a structure displaying all of the pages accessible from the current location in advance, as Hy+ does. But it is still not enough to distinguish the user's action. Even if the URL of the new page is in the URL set we get, it is not certain that the user clicked on one of anchors in the current page. For example, if the current page contains a link to a page that has been added to the Netscape's bookmarks. When CZWeb receives an AppleEvent from Netscape notifying that the page is visited, CZWeb does not know whether the user clicked on an anchor in the current page or selected an item from Netscape's Bookmarks.

6.1.4 Using Web Browsers

Web site management tools, such as WebMapper, allows the user to see the structure of a web site without viewing a single web page. If the user really wants to see the content of a particular page, he can do it by double click on the node, which launches his favorite web browser to display that page. The user can choose any popular web browsers. The function of web browsers is just as an HTML file viewer.

On the other hand, navigation tools depend on web browsers to access the Internet. They work only with a particular web browser. Early approaches also modify the browser for their special uses. This is very inconvenient to users.

CZWeb is in some place between those two extreme ends. CZWeb works with Netscape Navigator through its remote control API. We do not need to modify Netscape Navigator, but if Netscape modifies its API, we may have to upgrade our CZWeb too. It would be better to remove the dependence on a special web browser and allow the user choose to use his favorite web browser. This can be done by writing different functions to communicate with different Web browsers.

6.2 Future Work

From the results of evaluations and comparing CZWeb with other approaches and systems, we find that there are several areas we need to improve. We need to improve algorithms for better performance and add new features to satisfy users' needs.

6.2.1 Algorithm improvement

- Layout

  Our spring layout algorithm is too simple and does not work very well. There are still some overlaps among nodes and oscillation happens sometimes. We also need a better algorithm to find an appropriate location for new nodes.

- Speed

  The zoom algorithm and display algorithm work fine for a small number of

nodes. When the number of nodes increases, performance drops down quickly, especially when color is used for nodes' background, because it takes more time to update the view. Another algorithm which may affect the performance is the algorithm used to store and sort the URLs of visited web pages. Currently, the URLs are managed as a linked list and an inefficient algorithm is used to search the list when a new URL is received. We can replace the list with a hash table or use a more efficient searching algorithm.

6.2.2 New Features

- Automatically expanding view

  One feature we need to implement in CZWeb is automatically expanding a few levels from a selected node. This feature can help users make their decisions on where to go next and reduce cognitive overhead. Options should be provided for ways to expand, such as expand to a certain level and expand inside the current web site.

- Include content-based analysis

  CZWeb is a navigation-based approach, but it uses a structure-based approach to analyze the collected information. A content-based analysis approach is another important feature to organize information and should be implemented in CZWeb.

- Searching and filtering

  We need to add data filtering and searching facilities, which are useful to find and organize data when the number of nodes in the view increases. For example, displaying all of the web pages visited between one o'clock and two o'clock, finding and highlighting all of the web pages with "Home" as a sub-string in their URL.

- Annotation

The information currently recorded in page nodes is URLs and titles. URLs are just addresses to identify web pages. Titles tell us about the content of web pages briefly. Generally however, titles do not summarize the content appropriately. It would be better if CZWeb could provide a way to add a user's note to each node. The notes may be used as reminders about particular issues associated with web pages for the CZWeb map creator, or as some additional information to help people decide in advance exactly which of the documents in the map is worth visiting if the map is shared with other people.

- Handling frames

  HTML is being developed quickly. With frame tags, window of the browser can be divided into several regions or frames. Each of the frames in a window can show a different web page. Actions inside one frame can control web pages (each of them has a different URL) displayed on another frame. With this feature, Netscape can present information in a more flexible and useful fashion. But Netscape has not updated its API for remote control to handle frames. CZWeb cannot receive the URLs displayed in each of the frames if the URL displayed in Netscape's URL box does not change. There is no good solution to this problem right now.

- Handle more than one browser window

  Web browsers allow the user to use several windows at the same time. The current version of CZWeb does not distinguish the pages viewed by different browser windows. Only one page node is highlighted which represents the web page displayed on the active browser window.

6.2.3 Other work

- More Evaluations We have made efforts to evaluate the cluster-page version of CZWeb. The URL-structure version was developed later and has not been tested. If possible, we should design an experiment to compare the two versions. We can have two groups of subjects perform the same task and check the result.

- User interface consistency

  Both groups in the third evaluation revealed the same problem about the inconsistency between the user interface of Netscape and CZWeb. CZWeb uses double click to access visited pages while single click is used for selection. This scheme is consistent with the operating system but inconsistent with Netscape. Netscape uses single click to active links and access web pages. Users were confused with this, because they thought CZWeb was part of Netscape and they could use the same technique to access web pages. Although Netscape's way is inconsistent with the operating system, we cannot do any thing about that. But we can do something to make CZWeb's user interface consistent with Netscape's.

  The above is not a complete list. The Web and techniques used in the Web are being developed very quickly. There are many other areas we ought to look at to keep updated.

6.3 Conclusion

The World Wide Web has become an important and widely used resource and continues to attract increasing numbers of novice computer users. Because of this, it is crucially important to address its usability. We have shown one promising technique based on the Continuous Zoom algorithm to better support Web navigation. This technique was used to implement a prototype navigation aid, CZWeb, to help Netscape users.

In the CZWeb approach, the visited portion of the Web information space is recorded and organized as users navigate through it. The Web structure is represented with a simple view to help users understand it. The visual representation, a network with nested graphs, is built based on the physical URL structure so that it is intuitive and easy to understand. It also provides users with the ability to recognize objects by their relative positions and by the distinctive shapes that clusters of objects form. The ability to allow users to restructure the automatically created structure assists

in better understanding of the structure and relationship of the visited portion of the Web.

The Continuous Zoom algorithm provides a technique for visualizing large hierarchical information spaces. It displays parts of the information space in detail while simultaneously displaying the overall context. The extensions made to the Continuous Zoom algorithm in CZWeb add new functions to it and improve its performance.

We get some positive results from the evaluations. It shows that using a hierarchy to represent the web structure is consistent with users' mental model. Displaying a visual representation on Web's structure does help users understand the Web, localize their current location, and move around the Web. Facilities provided by CZWeb also help users go back to visited Web pages quickly. However, there are still areas we need to improve, such as making users feel as if they are traveling through the Web, building a clearer structure to represent the web, and continuing to improve the user interface.

Bibliography

[Andr94] K. Andrews and F. Kappe. "Soaring through hyperspace: A snapshot of Hyper-G and its Harmony client". *Proc. Eurographics Symp. and Workshop on Multimedia: Multimedia/Hypermedia in Open Distributed Environments*, Graz, Austria, June 1994.

[Ayue95] E.Z. Ayuers and J.T. Stasko. "Using Graphic History in Browsing the World Wide Web." Technical Report GIT-GVU-95-12, Georgia Inst. of Technology. URL: file://ftp.gvu.gatech.edu/pub/gvu/tech-reports/95-12.ps.Z

[Baec95] R. Baecker, J. Grudin, W. Buxton, and S. Greenberg, "Readings in Human-Computer Interaction: Towards the Year of 2000", Second Edition, Morgan Kaufmann Publishers, Inc., San Francisco, 1995, pp. 581.

[Bala96] V. Balasubramanian. "State of the Art Review on Hypermedia Issues and Applications", 1996
http://www.inf-wiss.uni-konstanz.de/Res/review/table_of_contents.html

[Bani96] E. Baniassad, I. Caven, H. S. Chin, Y. Sun, and P. Tan, "CZWeb Usability Study", *Course project report for CMPT-882, 96-2, Simon Fraser University*, 1996.
http://www.ensc.sfu.ca/GradStudents/put/personal/report_hci.ps

[Bede94] B. Bederson, J. Hollan, "Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics", *Proceedings of ACM UIST '94"*, ACM

Press, 1994.
See also: *http://www.cs.unm.edu/pad++/*

[Bede97] B. Bederson, J. Hollan, J. Stewart, D. Rogers, A. Druin, D. Vick, "A Zooming Web Browser", See also: *http://www.cs.unm.edu/pad++/*

[Bern94] T. Berners-Lee, R. Cailliau, A. Loutonen, H.F. Nielson, and A. Secret. "The World-Wide Web." *Communications of the ACM* 37, 8(August), 1994, p76-82.

[Bly 86] S.A. Bly and J.K. Rosenberg. "A comparison of tiled and overlapping windows." In *Proceedings of CHI'86*, ACM press, New York, 1986, pp. 101-106.

[Brow95] M.H. Brown and R.A. Shillner. "Deckscape: An experimental web browser." In *Proceedings of the Third International WWW Conference*, April 10-14 1995, Darmstadt, Germany.
http://www.igd.fhg.de/www/www95/papers/90/deckscape-final-v1/paper.html

[Bush45] V. Bush. "As We May Think," *The Atlantic Monthly*, July 1945.

[Card86] S.K. Card and T.P. Moran, "User Technology: From Pointing to Pondering." *Proc. ACM Conference on History of Personal Workstations*, pp. 183-198.

[Carp95] M. S. T. Carpendale, D. J. Cowperthwaite, and F. D. Fracchia, "3-Dimensional Pliable Surfaces: For the Effective Persentation of Visual Information". In *UIST: Proceedings of the ACM Symposium on User Interface Software and Technology,* pp. 217-226, 1995.

[Card96] S.K. Card, G.G. Robertson, and W. York, "The WebBook and the Web Forager: an Information Workspace for the World Wide Web." In CHI96, ACM Conf. on Human Factors in Software, ACM Press, New York, 1996

[Carr84] J. M. Carroll and R. L. Mack, "Learning to use a word processor: By doing, by thinking, and by knowing". In Thomas, J., and Schneider, M. (Eds.),

Human Factors in Computer Systems, Ablex, 1984, 13-51. Or In "Readings in Human-Comoputer Interaction: Towards the Year of 2000", Second Edition, written and edited by R. Baecker, J. Grudin, W. Buxton, S. Greenberg. Morgan Kaufmann, San Francisco, 1995, pp. 698-717.

[Carw69] S. Carmody, W. Gross, T. Nelson, D. Rice, and A. van Dam, "A Hypertext Editing System for the 1960." In *Pertinent Concepts in Computer Graphics,* M. Faiman and J. Nievergelt, eds. University of Illinois Press, 1969, pp. 291-330.

[Catl95] L. D. Catledge and J. E. Pitkow, "Characterizing Browsing Strategies in the World-Wide Web", In *Proceedings of the Third International WWW Conference*, April 10-14 1995, Darmstadt, Germany. http://www.igd.fhg.de/www/www95/papers/80/userpatterns/UserPatterns.Paper4.formatted.html

[Coll87] G. Collier, "Thoth-II: Hypertext with explicit semantics." In *Hypertext '87 Papers* (Chapel Hill, N.C., Nov. 1987). University of North Carolina, Chapel Hill, 1987, pp. 269-287.

[Coll95] G. Collaud, J. Dill, C. V. Jones and P. Tan, " A Distorted-View Approach to Assisting Web Navigation", *New Paradigms in Information Visualization and Manipulation, in conjunction with CIKM'95* Baltimore, December 1995.

[Coll96] G. Collaud, J. Dill, C.V. Jones and P. Tan, "The Continuously Zoomed Web – A Graphical Navigation Aid for WWW", *IEEE Visualization'96 Late Breaking Hot Topics Papers*, Ebert, D.S.(Ed), pp.1-3, 1996.

[Conk87] J. Conklin, "Hypertext: An Introduction and Survey." IEEE Computer 2(9), (Sept. 1987), 17-41.

[Conk88] J. Conklin and M. Begeman. "gIBIS: A hypertext tool for team design deliberation." *ACM Trans. OFF. Info. Syst.* 6, 4 (Oct. 1988), 303-331.

[Cour96] Course CMPT-882, offered by Dr. Tom Calvert in School of Computing Science of SFU and Dr. Kelly Booth in Department of Computing Science of UBC in spring semester 1996.

[Cubr97] D. Cubranic, R. Hwang, J. Madar, and M. Mizuguchi, "Analyzing CZWeb", *Course project report for CPSC 533B (UBC) CMPT 873 (SFU), 97-1*, 1997. http://www.sfu.ca/ mmizuguc/report.ps.gz

[Desa94] B.C. Desai, and S. Swiercz. "WebJournal: Visualization of a Web Journey." 1995
ftp://ftp.cs.concordia.ca/pub/bcd/WebJournal/

[Dill93] A. Dillon, C. McKnight, and J. Richardson. "Space - the final chapter or why physical representations are not semantic intensions." In C. McKnight, A. Dillon and J. Richardson, Eds.. *Hypertext - a psychological perspective.* Ellis Horwood, New York, 1993, pp. 169-191.

[Dill95] J. Dill, L. Bartram, A. Ho, and F. Henigman, "A Continuously Variable Zoom for Navigating Large Hierarchical Networks", *Proceedings of the IEEE Conference on Systems, Man and Cybernetics SMC-94*, 1994, pp. 386-390.

[Dome94] P. Domel. "Webmap - a graphical hypertext navigation tool." In *Proceedings of the Second International World Wide Web conference'94: Mosaic and the Web*, Chicago, USA, October 1994.
http://www.ncsa.uiuc.edu/SDG/IT94/IT94Info-old.html

[Fein88] S. Feiner. "Seeing the forest for the trees: Hierarchical display of hypertext structure." In *Proceedings of the Conference on Office Information System* (Palo Alto, Calif., Mar. 23-25, 1988). ACM, New York, 1988, pp. 205-212.

[Fish97] B. Fisher, M. Agelidis, J. Dill, P. Tan, G. Collaud, C. Jones, "CZWeb: Fish-eye Views for Visualizing the World-Wide Web", *HCI International 97.*
http://palette.ecn.purdue.edu/ salvendy/hci97

[Foss88] C. Foss. "Effective browsing in hypertext systems." In *RIAO'88 Conference Proceedings* (Cambridge, Mass., Mar. 1988). MIT, Cambridge, Mass., 1988, pp. 82-98.

[Fowl96] R. Fowler, A. Kumar, J. Williams. "Visualizing and Browsing WWW Semantic Content", 1996.
http://bahia.cs.panam.edu/info_vis/doc_explorer/cetac96.html

[Furn86] G.W. Furnas. "Generalized fisheye views." In *Proceedings of the CHI'86 Conference*, Boston, MA, 1986, pp. 16-23.

[Game] http://www.ensc.sfu.ca/GradStudents/put/personal/THA.html

[Gers95] N Gershon, J. LeVasseur, J Winstead, J. Croall, A. Pernick, and W. Ruh. "Visualizing Internet Resource." In *proceedings of the IEEE Symposium on Information Visualization '95*, 1995.

[Hala87] F.G. Halasz, T.P. Moran, and R.H. Trigg. "Notecards in a nutshell." In *Proc. Human Factors in Computing Systems (CHI'87)*, pages 45-52. ACM Press, 1987.

[Hasa95] M.Z. Hasan, A.O. Mendelzon, and D. Vista. "Visual Web Surfing with Hy+." *Proc. CASCON'95*, Toronto, 1995, pp. 218-227.
See also http://www.db.toronto.edu:8020/webvis.html

[Hend95] R.J. Hendley, N.S. Drew, A.M. Wood, R. Beale. "Narcissus: Visualizing Information." In *Proceedings Information Visualization'95 (October 30-31, 1995, Atlanta, Georgia, USA*. IEEE Computer Society Press. Los Alamitos, California, 1995. pp. 90-96.

[Java96] http://www.javasoft.com/applets/applets/GraphLayout/example1.html

[Kaeh88] C. Kaehler, "HyperCard Power: Techniques and Scripts". Addison-Wesley, reading, Mass., 1988, pp. 18-19.

[Kahn73] D. Kahneman. *Attention and Effort*, Prentice-Hall, Englewood Cliffs, NJ, 1973.

[Kamp95] T. Kamps and J. Kleinz. "Constraint-Based Spring-Model Algorithm for Graph Layout". In *Proceedings of Graph Drawing 1995*, Passau, Germany, September 20-22, 1995. Lecture Notes in Computer Science, Springer Verlag, 1995.

[Kons96] K. Guericke. "What is VRML?"
http : //livedv.com/Whitepapers/VRML.html

[Krol95] E. Krol, *The Whole Internet: User's Guide & Catalog*, Second Edition, O'Reilly & Associtates, Inc., 1995. pp. 30-32.

[Lamp95] J. Lamping, R. Rao, and P. Pirolli. "A focus+context technique based on hyperbolic geometry for visualizing large hierarchies." In *Proceedings of the ACM SIGCHI conference on Human Factors in Computing Systems*, (ACM, May 1995).

[Maar96] Y. Maarek, and I. Shaul, "Automatically Organizing Bookmarks per Contents", In *Proceedings of the Fifth International World Wide Web Conference*, Paris, France, May 1996.

[Marc90] A. Marcus, "Principles of Effective Visual Communication for Graphical User Interface Design." UnixWorld, August 1990, 107-111; September 1990, 121-124; and October 1990, 135-138 (revied and reformatted).

[Meyr86] B.J. Meyrowitz, K.E. Smith, and L.N. Garrett. "Intermedia: Issues, strategies and tactics in the design of a hypermedia document system." In *Proc. ACM Conference on Computer-Supported Cooperative Work (CSCW'86)*, pp. 163-174. ACM, 1986.

[Muk195] S. Mukherjea, J.D. Foley, S. Hudson, "Visualizing Complex Hypermedia Networks through Multiple Hierarchical Views." In Proc. of CHI'95, ACM Press, 1995.

[Muk295] S. Mukherjea, J.D. Foley, "Visualizing the World-Wide Web with the Navigational View Builder." In *Proceedings of the Third International World Wide Web Conference*, April 10-14 1995, Darmstadt, Germany. http://www.igd.fhg.de/www/www95

[Nels65] T. Nelson. "A File structure for the Complex, The Changing and The Indeterminate". ACM 20th National Conference, 1965.

[Netc96] http://www.netcarta.com/

[Nets94] http://home.mcom.com/newsref/std/mac-remote-control.html

[Neuw87] C. Neuwirth, D. Kauffer, R. Chimera, and G. Terilyn. "The Notes Programs A hypertext Application for Writing from Source Texts." In *Proceedings of Hypertext'87 Conference*", Pages 121-135, Chapel Hill, NC, Novermber 1987.

[Niel90] J. Nielsen. "The art of navigation through hypertext." *Communication of the ACM* **33**, 3 (March 1990), pp. 296-310.

[Niel95] J. Nielsen. "Multimedia and Hypertext: The internet and beyound". Boston, Academic Press, 1995.

[Nigu97] G. Niguma(1997). "Concept Mapping in a Multimedia, World Wide Web Environment", MSc thesis, Simon Fraser University, Apr. 1997.

[Pesc95] Mark Pesce. *VRML: Browsing and Building Cyberspace*, New Riders, August 1995.

[Pitk94] J.E. Pitkow, K.A. Bharat. "WebViz: A Tool for WWW Access Log Analysis." *Technical Report of Georgia Institute of Technology*, 1994. http://www.cc.gatech.edu/gvu/research/techreports94.html

[Pitk96] J.E. Pitkow and C.M. Kehoe, "Emerging Trends in the WWW User Population". *Communications of the ACM*, **39:6** pp. 108-108, 1996.

[Proj96] *http://product.info.apple.com/pr/press.releases/1996/q4/960918.pr.rel.internet.html*

[Rao 90] U. Rao and M. Turoff. "Hypertext Functionality: A Theoretical Framework". *International Journal of Human-Computer Interaction*, 1990.

[Rivl94] Ehud Rivlin, Rodrigo Botafogo, and Ben Shneiderman. "Navigating in Hyperspace: Designing a Structure-based Toolbox." *Communications of the ACM*, Vol. **37**, No.2, February 1994.

[Robe91] G.G. Robertson and J.D. Mackinlay and S.K. Card. "Cone trees: Animated 3-D visualizations of hierarchical information". In *Proceedings of the ACM SIGCHI conference on Human Factors in Computing Systems*, (ACM, April 1991).

[Robe93] G.G. Robertson and J.D. Mackinlay. "The document lens." In *Proceedings of the ACM Symposium on User Interface Software and Technology*, ACM Press, November 1993.

[Shne87] B. Shneiderman. "User interface design and Evaluation for an Electronic Encyclopedia". *Proceedings of the 2nd International Conference on Human-Computer Interaction*, North-Holland, 1987.

[Shne87] B. Shneiderman. "User interface design for the Hyperties electronic encyclopedia". *Proc. ACM Hypertext'87 Conf.* (Chapel Hill, NC, 13-15 Novermber), 189-194.

[Shne92] B. Shneiderman. "Tree visualization with tree-maps: 2-d space-filling approach." *ACM Transactions on Graphics*, 11(1):92-99, January 1992.

[Smit88] J. Smith and S. Weiss. "An Overview of Hypertext". CACM July 1988.

[Stor95] M. Storey and H. Müller, "Manipulating and documenting software structures using shrimp views". In *Proceedings of the 1995 International Conference on Software Maintenance (ICSM '95)* Opio (Nice), france, October 16-20, 1995.
http://www.rigi.csc.uvic.ca/rigi/people/mstorey/publications.html

[Stor97] M. Storey K. Wong and H. Müller, "Rigi: A Visualization Environment for Reverse Engineering." In *Proceedings of the International Conference on Sottware Engineering (ICSE'97)*, Boston, USA, pp. 606-607, May 17-23, 1997.

[Tayl86] J. M. Taylor and G. M. Murch, "The Effective Use of Color in Visual displays: Text and Graphics Applications." *Color Research and Application*, Vol. *11*, Supplement 1986, pp. S3-s10.

[Thur95] M. Thruing, J. Hannemann, and J.M. Haake. "Hypermedia and Cognition: Designing for Comprehension". *Commun. ACM* 38:8, pp. 57-66, August, 1995.

[Utti89] K. Utting and N. Yankelovich. "Context and Orientation in Hypermedia Networks". *ACM Transactions on Information Systems*, Vol. **7**, No. 1, January 1989, pp. 58-84.

[VanD88] A. Van Dam. "Hypertext '87 Keynote Address". *Communications of the ACM*, 31(7), 1988, pp. 887-895.

[Walk97] R. Walker, X. Hu, and V. Rajamanickam, "CZWeb Patterns of Use by Novices: CPSC 533B/CMPT 873 Term Project", *Course project report*, 1997.
http : //www.ensc.sfu.ca/GradStudents/xhu/personal/czweb_hci.ps

[WebA96] http://www.incontext.com/products/wareview/review1.html

[Wise95] J.A. Wise et al., "Visualizing the Nonvisual: Spatial Analysis and Interaction with Information from Text Documents," *Proc. info. Vis. Symp. 95, N.*

Gershon and S.G.Eick, eds., IEEE Computer Society Press, Los Alamitos, Calif., 1995, pp. 51-58.

[Zizi94] Mountaz Zizi, Michel Beaudouin-Lafon, "Accessing Hyperdocuments through Interactive Dynamic Maps." *ECHT '94 Proceedings*, Sept. 1994.