INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

ENABLING THE REUSE OF WORLD WIDE WEB DOCUMENTS IN TUTORIALS

by

David Johnson

A dissertation submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

University of Washington

1997

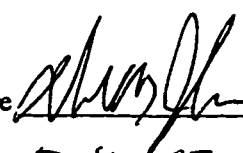
Approved by Steven L. Tanimoto
Chairperson of Supervisory Committee

Program Authorized to Offer Degree Computer Science and Engineering

Date May 16, 1997

Doctoral Dissertation

In presenting this dissertation in partial fulfillment of the requirements for the Doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to University Microfilms, 1490 Eisenhower Place, P.O. Box 975, Ann Arbor, MI 48106, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature 

Date 5-16-97

University of Washington

Abstract

ENABLING THE REUSE OF WORLD WIDE WEB DOCUMENTS IN TUTORIALS

by David Johnson

Chairperson of the Supervisory Committee: Professor Steven Tanimoto
Department of Computer Science and Engineering

The World Wide Web is a rich source of material that could be used for educational purposes. Advantages of reusing these existing materials are avoiding the duplicated effort of creating new versions of material that is already in existence, using up-to-date information, and letting the student see the original source materials. This thesis identifies and describes some of the problems that arise when reusing existing materials in tutorials. The thesis then describes Wtutor (Web Tutor), a prototype system for exploring solutions to these problems. Wtutor allows a tutorial author to create a tutorial as a set of paths through the network of documents on the Web and to include assessment points along the trail. Wtutor monitors the student's progress through the tutorial and provides navigational guidance to the student as well as feedback regarding the student's performance on the assessment points. Because the retrieval of Web documents can be slow and unreliable, Wtutor implements a variation of just-in-time prefetching, which allows Wtutor to retrieve most Web documents within tutorials before the student is ready to begin the documents, so student-experienced delays are greatly reduced and frequently completely eliminated. Reusing documents that are owned by authors other than the tutorial author means that the tutorial author does not have control over the content of the included documents. The owners of the documents may make changes to those documents that may or may not affect the appropriateness of the documents for inclusion in the tutorial. In Wtutor, document signatures have been designed and used to confirm that documents have not changed in ways that make them no longer appropriate for inclusion

in the tutorials. Wtutor provides an environment for both authoring and taking World Wide Web-based tutorials. Future work in this area will focus on providing an enhanced authoring interface and on developing a more portable implementation of the functionality that Wtutor provides.

TABLE OF CONTENTS

LIST OF FIGURES iv
LIST OF TABLES vi
INTRODUCTION 1
A SAMPLE WEB-BASED TUTORIAL 3
THE RETRIEVAL PROBLEM 8
THE VALIDATION PROBLEM 12
OVERVIEW OF THE DISSERTATION 13
CHAPTER 2: RELATED WORK 14
THE FILTERING AND NAVIGATION PROBLEMS 15
TRAILS, TOURS AND PATHS 16
TOURS ON THE WORLD WIDE WEB 19
CHAPTER 3: BASIC TUTORIAL SUPPORT 24
TUTORIAL MAPS 24
TMOSAIC 27
WTUTOR 28
AUTHORING TUTORIALS 31
TOOLS FOR GENERAL TRANSITION PROBLEMS IN WWW BASED TUTORIALS 31
STUDENT FEEDBACK 33
CHAPTER 4: SOLVING THE RETRIEVAL PROBLEM 37
CONCEPTUAL MODEL OF PREFETCHING 40
IMPLEMENTATION OF WWW PREFETCHING 47
PREFETCHING RESULTS 52
UNAVAILABLE DOCUMENTS 57
CHAPTER 5: SOLVING THE VALIDATION PROBLEM 59

CHANGES TO DOCUMENTS 59
CONCEPTUAL MODEL OF VALIDATION 63
DESIGN OF SIGNATURES 64
Distributed systems 68
Software maintenance 69
Information retrieval 70
WTUTOR SIGNATURES 71
Paragraph checksums 72
List of headings in document 73
Keyword based signatures 74
WTUTOR SIGNATURE COMPARISON METHOD 74
Default weights 76
Analysis of Wtutor signature comparison method 78
TESTING SIGNATURE COMPARISONS RESULTS 80
Controlled changes to selected documents 80
Document life cycle 86
Sample tutorial 89
Author volunteered examples 91
CHAPTER 6: EVALUATION 93
WTUTOR MODEL OF STUDENT INTERACTION 94
PLATFORM AND BROWSER DEPENDENCE 98
PREFETCHING AND CACHING ALGORITHMS 99
TUTORIAL AUTHORING INTERFACE 101
FUTURE WORK 102
Enhancements to basic tutorial support 103
Retrieval and prefetching issues 103
Enhancements to validation 104

BIBLIOGRAPHY 107

LIST OF FIGURES

Number *Page*

Figure 1 Introduction to the Martin Luther King tutorial 4

Figure 2 Biography document of Martin Luther King tutorial 5

Figure 3 Chronology document of Martin Luther King tutorial 6

Figure 4 Civil Rights document of Martin Luther King tutorial 7

Figure 5 Letter from a Birmingham Jail document from Martin Luther King tutorial 9

Figure 6 "I Have a Dream" document from Martin Luther King tutorial 10

Figure 7 Sample page from Walden's Paths path server 21

Figure 8 Wtutor environment 25

Figure 9 Sample tutorial map 25

Figure 10 Tmosaic Tutor menu 28

Figure 11 Tmosaic Load Tutorial Map popup window 29

Figure 12 Fill-out form for creating assessment items 34

Figure 13 Sample assessment item with Wtutor feedback 35

Figure 14 Reading tutorial without prefetching 41

Figure 15 Reading tutorial with perfect prefetching 41

Figure 16 Reading tutorial with retrieval times greater than visitation times 43

Figure 17 Adjusting for retrieval times greater than visitation times 44

Figure 18 Tutorial map with alternative nodes 46

Figure 19 Using visitation time as a window for prefetching 50

Figure 20 Impact of high visitation time estimates on just-in-time policy 51

Figure 21 Visitation and retrieval times for selected URLs 54

Figure 22 Sample Wtutor signature 72

Figure 23 Document signature distance function 75

Figure 24 Default distance function weighting factors 77
Figure 25 Sample document A 77
Figure 26 Sample document B 77
Figure 27 Document signature distance function 78
Figure 28 Selected WWW documents 81
Figure 29 Distances from original documents to modified versions 84
Figure 30 Versions of computer document ordered by distance measure 86
Figure 31 Output of diff comparison of version 0 and version 1 87
Figure 32 Output of diff comparison of version 1 and version 2 88
Figure 33 Output of diff comparison of version 2 and version 4 89

LIST OF TABLES

Number *Page*

Table 1 Retrieval and visitation times for selected URLs 53

Table 2 Selected WWW document characteristics 82

Table 3 Controlled changes applied to selected WWW documents 83

Table 4 Distance measures for versions of Wtutor document 89

Table 5 Volunteered WWW document distance measures 92

ACKNOWLEDGMENTS

The author wishes to thank a number of people, without whom this work would not have been possible. My deepest thanks go to my advisor, Professor Steve Tanimoto, for pushing me to expand my horizons and for having the patience to stick with me throughout this effort. My thanks also to my committee members, especially Professor William Winn and Professor David Farkas for their early feedback and suggestions and their willingness to send students to help with testing and evaluation.

I would like to thank U S WEST for the fellowship that supported my research. John Agnew, John Snyder and Tom Overton provided the financial and administrative support as coordinators of the Advanced Technical Education Program and I am very grateful for their efforts. Special thanks go to Claudia Burke, Bonnie Klimek, Maggie Moore and Dave Kohler, who as my managers provided me with encouragement, a little prodding, and the time needed to pursue my research. My thanks also to the NCSA for making the Mosaic source code available.

Many of my fellow students helped with this work. I would like to thank Joe Sherman for his assistance with implementing the Lisp sockets connection, and thanks to Joe, Suzanne Bunton, Lauren Bricker and Rich Segal for lots of brainstorming and encouragement. The gang at the Chateau deserves thanks for volunteering to be guinea pigs. I would also like to thank Sara Kim, who put in a great amount of work in trialing Wtutor and building the Opera tutorial.

Finally, I would like to thank Kathy for her support, understanding and flexibility.

INTRODUCTION

The World Wide Web (WWW) is a rich source of material that can be used for educational purposes. The research presented in this dissertation focuses on enabling the incorporation of existing WWW materials into tutorials. One way of reusing materials found on the Web would be to extract the relevant material from the source documents and construct new documents for the tutorial. Another approach would be to make copies of the selected materials and then modify those documents just enough to add navigational links to create the flow through the tutorial. The approach used in this research is to use the original source documents within the tutorial. The assumption that the original source documents will be used within the tutorial will be called the "Original source document" assumption. The Original source document assumption prevents possible problems with intellectual property rights but presents some other problems that have implications for what features the tutorial system must provide.

Reusing existing materials requires a means of adding structure that makes the materials fit within the tutorial flow. Since the tutorial author may not be the owner of the existing material, the method of structuring the materials must not require altering the content of the included documents. When the tutorial author is not the owner of some materials within a tutorial, the possibility exists that those materials may change without the tutorial author's knowledge. This means that some method of validation is needed to

ensure that materials included in the tutorial have not changed too much. Finally, the distributed nature of the Web leads to possible delays in retrieving documents; the reuse of existing materials requires techniques for dealing with delays or even the unavailability of documents.

There are a number of reasons for reusing materials. The primary reason is that if a tutorial author can reuse existing material, the author is saved the time and effort needed to create original material that merely duplicates what is already available. This is especially true in the cases where the existing material is exceptional—very well written, or written by an acknowledged expert in a field, or provided by the sole source (as in photos from space provided by NASA). Another potential advantage is that the original author may regularly update the contents of materials. In cases where the materials cover rapidly changing fields or provide current data about real-world events, this saves the tutorial author from having to frequently update the materials. Reusing existing materials allows the tutorial author to focus on adding value by organizing and commenting on the existing materials and only creating new material where needed, or where the tutorial author has expertise or significant new contributions to make.

The World Wide Web provides a vast store of globally available information from a variety of sources. Government agencies such as the National Institute of Health, NASA, and the Smithsonian operate multiple Web servers that allow access to data, reports, art and analysis. Most universities and colleges have Internet access and these institutions, departments, faculty, staff and students have set up Web servers to provide

information about the schools and their research. Organizations such as CERN make specifications and proposals available via the Web. Businesses and individuals operate Web sites for self-promotional purposes or for love of a subject. All of these different resources can be used as the source of materials for inclusion in tutorials.

Using existing materials from the WWW for educational purposes presents a number of problems that must be dealt with. The first problem is the *filtering problem*: with all the material out there on the Web, how does the student know which materials are relevant to the chosen subject area? And of those relevant materials, which best present the subject at a level appropriate for the student? Once the set of appropriate materials has been selected the student is confronted with the *navigation problem*: given the set of appropriate materials, where should the student start? In what order should the student view those materials? When is the student finished? The next problem is the *retrieval problem*: are the materials available at the time the student is taking the tutorial? How quickly can the materials be retrieved? Will the student's train of thought be broken by a long wait for the retrieval of the materials? Finally, assuming the materials can be retrieved, there is the *validation problem*: is the content of the retrieved materials still appropriate for the tutorial? Or has the material been changed so much that it is no longer appropriate for inclusion in the tutorial?

A SAMPLE WEB-BASED TUTORIAL

In this section, a sample Web-based tutorial is presented. The sample illustrates how an author could use materials from the Web as an integral part of the tutorial, and

also demonstrates some of the problems encountered. The sample tutorial uses materials from a number of sites to explore the life of Dr. Martin Luther King.

In most cases, the tutorial author will likely create an introduction to the tutorial. This allows the tutorial author to establish the context of the tutorial and provides a logical starting point for the student. The initial document of the sample tutorial is shown in Figure 1. Next up on the tour are two documents from the Martin Luther King, Jr., Papers Project at Stanford University, which provides a wealth of research material about Dr. King online. The first of these documents is a brief biography of Dr. King; the first

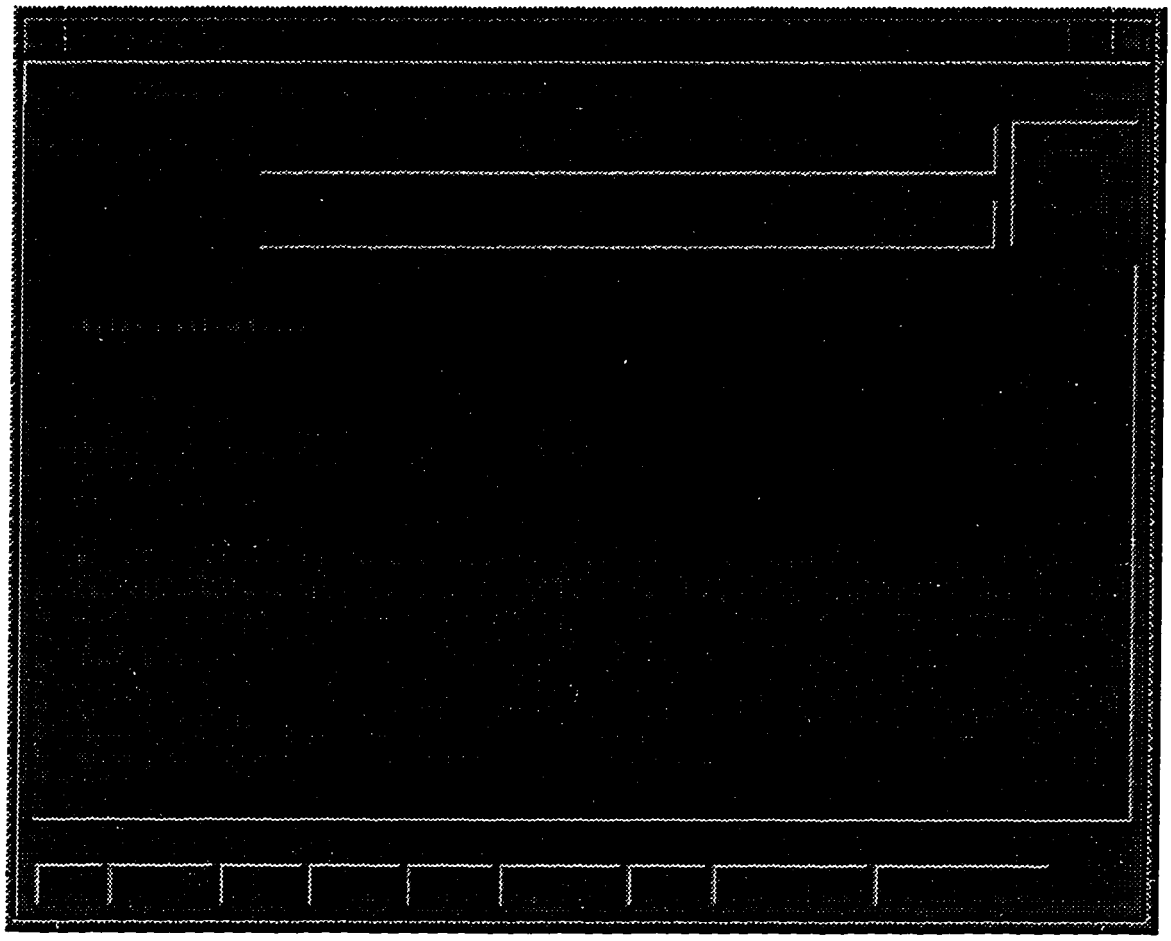


Figure 1 Introduction to the Martin Luther King tutorial

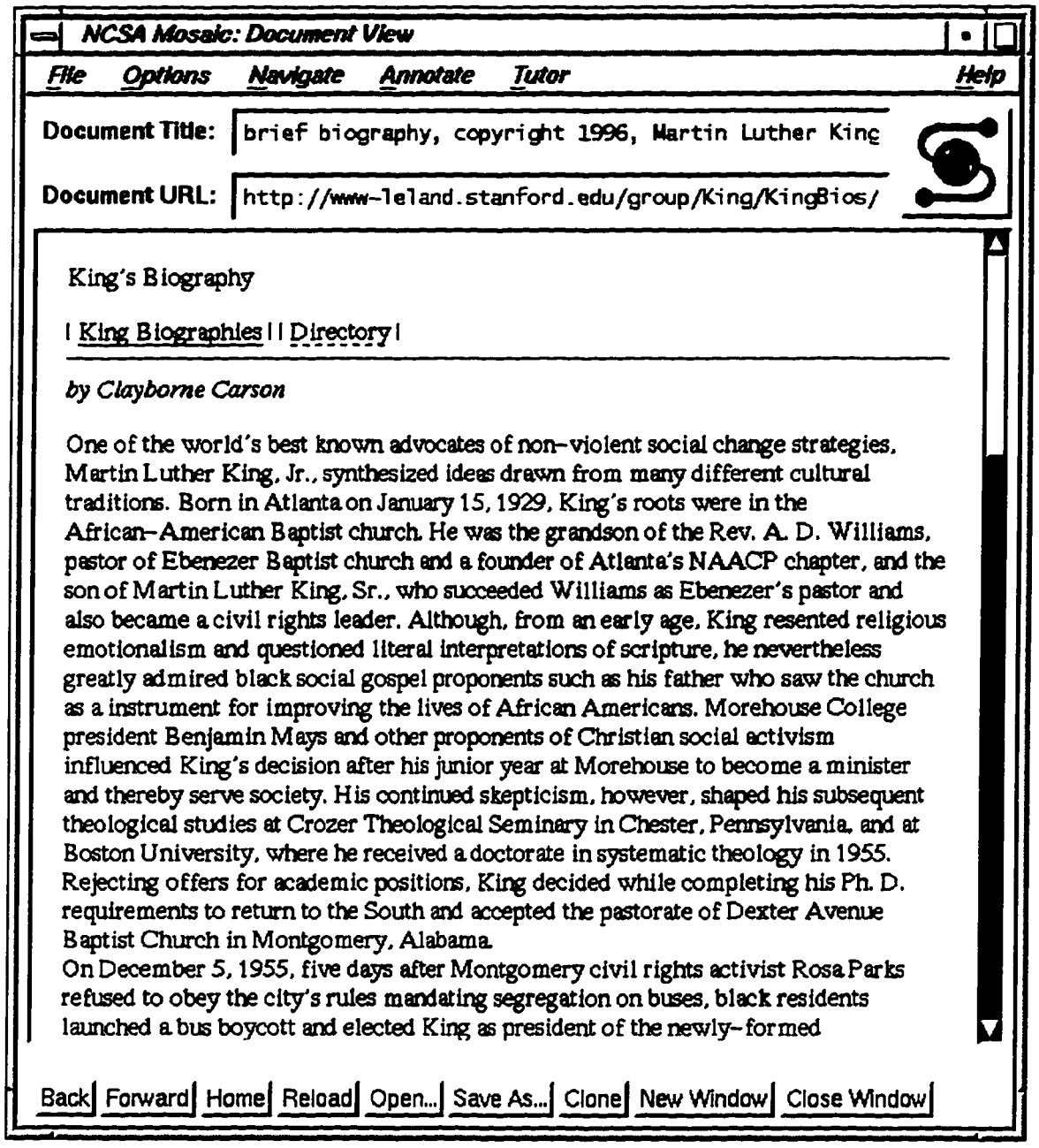


Figure 2 Biography document of Martin Luther King tutorial

page of this document is shown in Figure 2. The second of these two documents is a chronology which shows the timeline of significant events in Dr. King's life; the first page of the chronology document is shown in Figure 3.

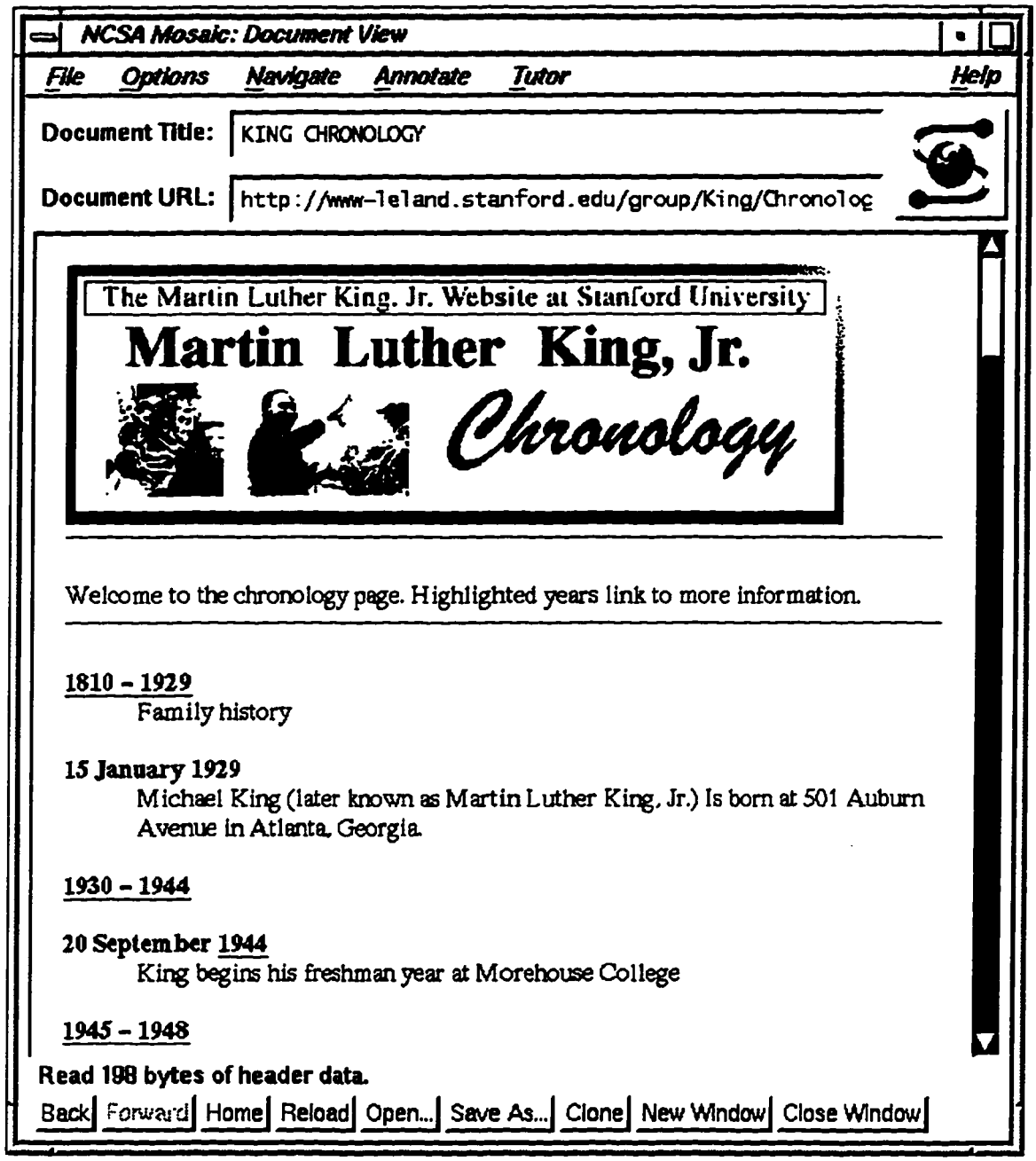


Figure 3 Chronology document of Martin Luther King tutorial

These first three documents illustrate the need for a path mechanism for the World Wide Web. Since the tutorial author created the Introduction, the author was able to put in a link from that document to the biography document. However, the tutorial author is

not the owner of the biography document and therefore cannot update it to add a link to the chronology document. Even though the biography and chronology documents come from the same server, there is no link from the biography to the chronology. For the tutorial to make a direct transition from the biography document to the chronology document requires a path mechanism so that the tutorial author can specify that the chronology should follow the biography, and to lead the student from one to the next.

Following the chronology is another transitional document created by the tutorial author. This document introduces the civil rights section of the tutorial. The civil rights document is shown in Figure 4. As before, the tutorial author cannot add a link from the

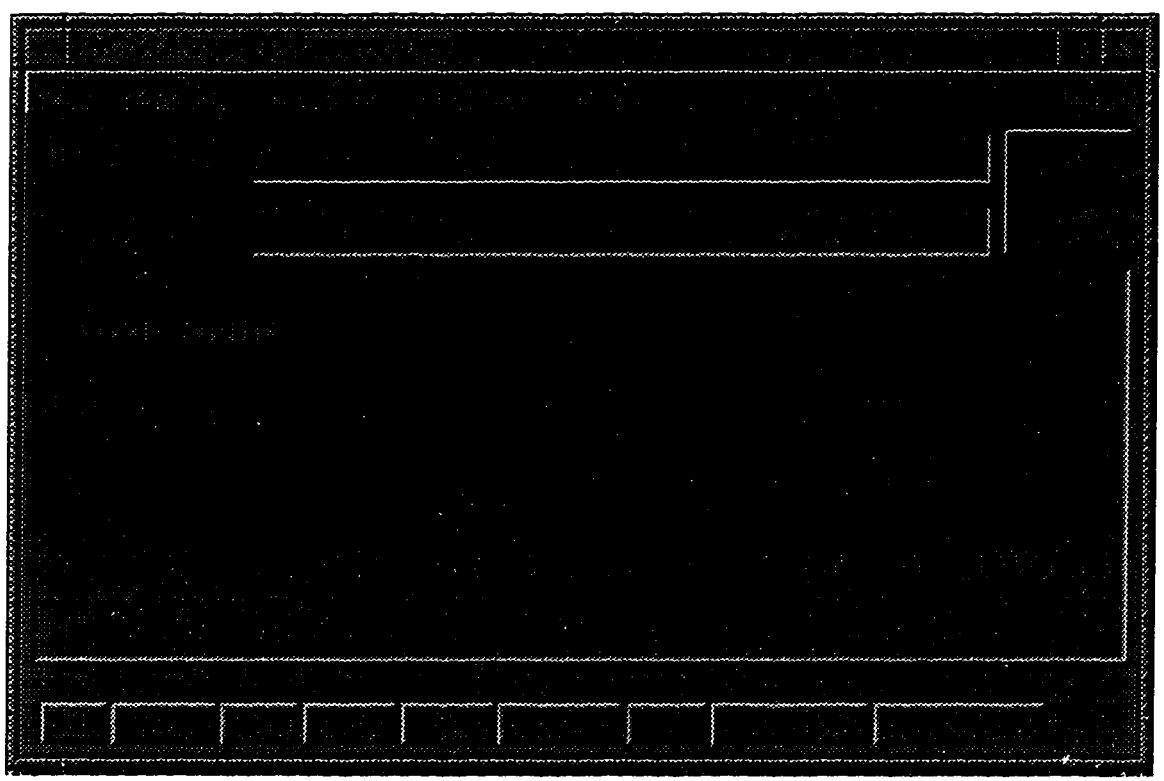


Figure 4 Civil Rights document of Martin Luther King tutorial

chronology document to the civil rights document, so a path mechanism is required for the tutorial to flow from the chronology document to the civil rights document. The civil rights document introduces the next two documents in the tutorial. Each of these next two documents is provided by a historical archive at another university. The text of Dr. King's Letter from a Birmingham Jail is found on the Historical Text Archive at Mississippi State University. The first page of the text of Dr. King's letter is shown in Figure 5. The text of Dr. King's "I Have a Dream" speech is provided by Steve Thoemke at San Francisco State University; the first page of this speech is shown in Figure 6.

The sample tutorial contains many more documents from the sources already mentioned as well as other sites on the World Wide Web. This short sample tour gives an indication of the richness of the Web as a source of material. It also illustrates why a path mechanism is needed: to lead a student through a set of unlinked but related materials in a meaningful, logical sequence. The path mechanism provides a way for the tutorial author to eliminate the filtering and navigation problems for the student. However, the student is still confronted with the retrieval and validation problems.

THE RETRIEVAL PROBLEM

With self-contained hypertext documents, the hypertext author does not have to worry about retrieval of the document contents. In general, retrieval is an all-or-nothing proposition: if the system that the hypertext document resides on is working, the system will be able to retrieve the document and the retrieval time is dependent on the speed and performance of the system. Because the WWW is a distributed hypertext system, the

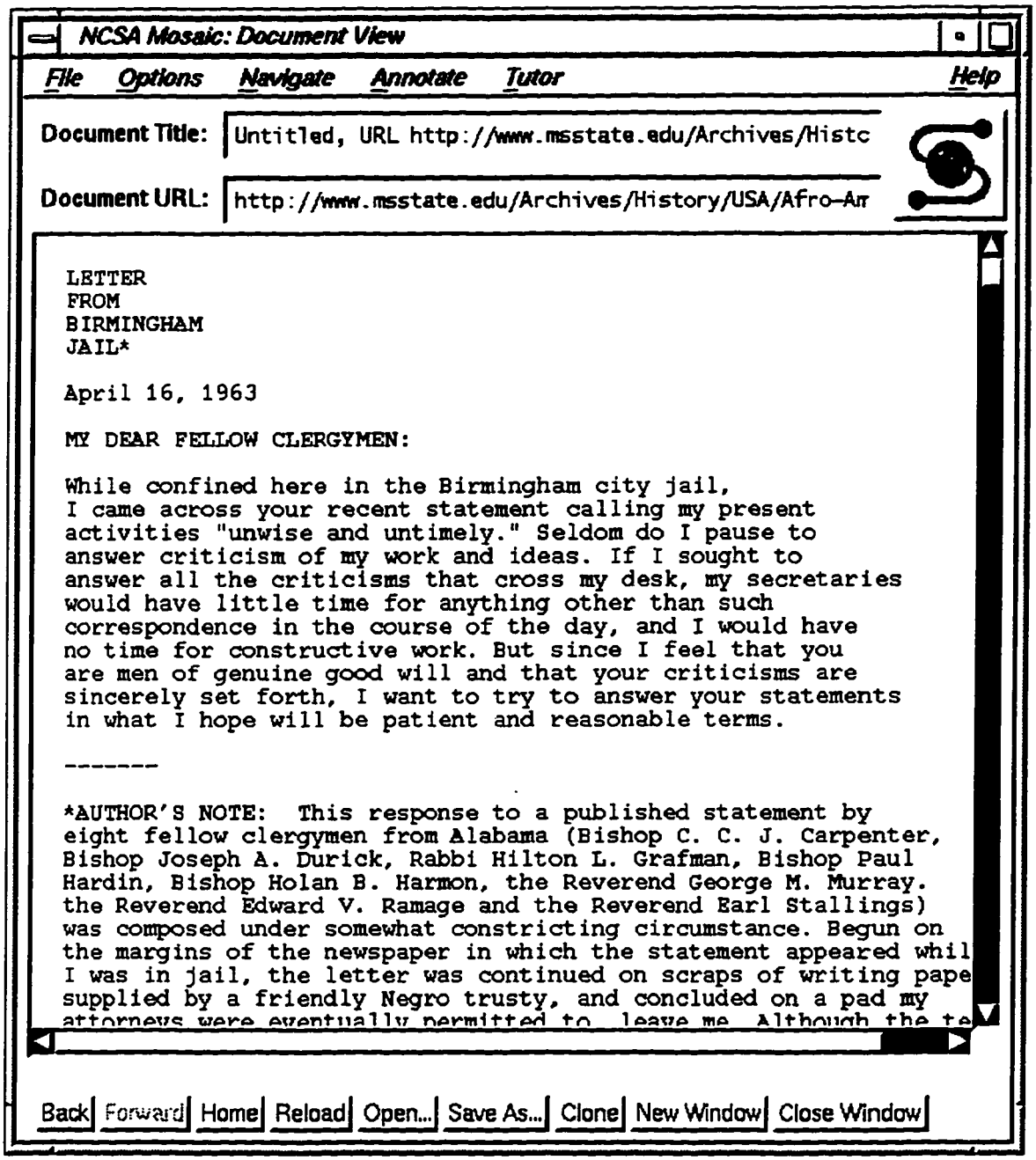


Figure 5 Letter from a Birmingham Jail document from Martin Luther King tutorial

tutorial author can no longer depend on prompt retrieval of all components of the tutorial when a student is taking the tutorial.

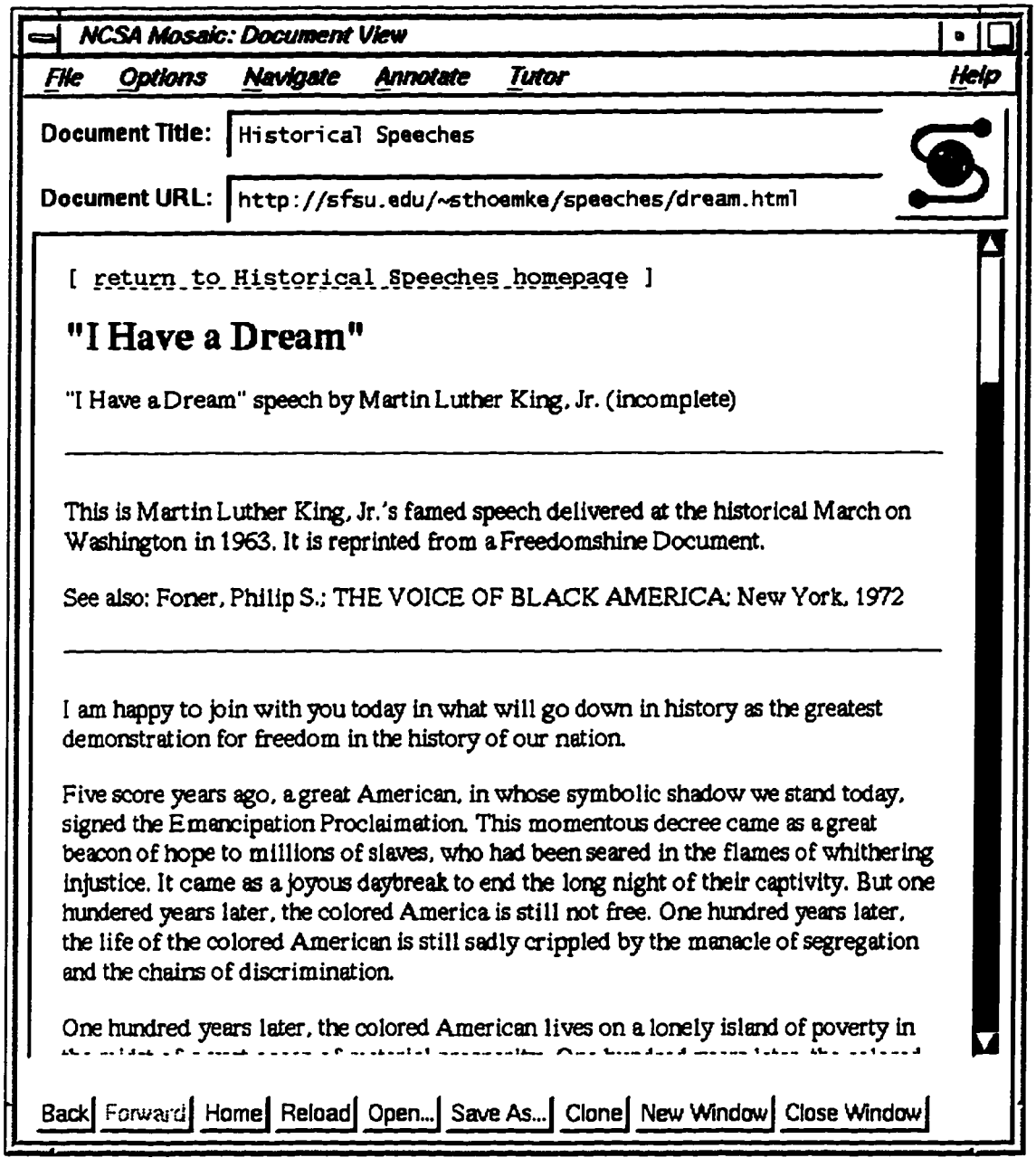


Figure 6 "I Have a Dream" document from Martin Luther King tutorial

A number of factors can result in delayed retrieval of a Web document—the server the document resides on may be slow, or have a heavy work load, or the network connection between the client and server may have a slow link or be heavily loaded. And these circumstances can be temporary or permanent. Slow response to retrieval requests is a frequent phenomenon on the Web. In a tutorial situation, slow response can be significantly disruptive to the student's chain of thought. It may cause frustration and irritation with the tutorial.

A Web-based tutorial system should have some means of avoiding student-experienced delays as the student progresses through the tutorial. Delays can be avoided if documents are prefetched; that is, retrieved before the student requests them. The use of paths allows for informed prefetching of documents because the path identifies the URLs which will be viewed and the sequence in which they will be viewed. Prefetching requires balancing goals with conflicting requirements: avoiding student-experienced delays, minimizing excess bandwidth consumption on the Internet, and effectively utilizing local cache space.

In some cases, documents will not be available at all. A Web server may be down for maintenance or because of problems. A Web server may be overloaded with requests and therefore unable to deliver a requested document. There may be network connectivity problems. Any of these could lead to failed requests for documents.

An author should take reliability of a document (or its associated Web server) into account when selecting URLs for inclusion in a tutorial. If a document is on a server that

is frequently down, students taking the tutorial will experience frustration as well as interruptions in the learning process. However, the tutorial system must make allowances for those circumstances in which documents are unavailable.

THE VALIDATION PROBLEM

As noted earlier, reusing existing materials has a number of benefits. However, the fact that someone other than the tutorial author has ownership of those materials can present a problem: the owner of the materials may make changes that make the material no longer appropriate for inclusion in the tutorial. The dilemma lies in the fact that some changes are desirable, while others are not.

For instance, if an author corrects some mistakes in a document, that is a positive change that would be welcome. Similarly, if an author adds new material that reflects new developments in a field, that is probably a positive and welcome change. However, if the new material is significantly longer than the original, or goes into much greater depth, or branches out into a different topic, then the change may not be a positive one with respect to the tutorial.

When a tutorial author selects materials for inclusion in a tutorial, the author must have a target audience in mind and select materials appropriate for that target audience. If a tutorial is targeted to a junior-high audience, then the inclusion of graduate school research papers is probably inappropriate. The tutorial author needs some assurance that when a student is taking the tutorial that documents selected for inclusion in the tutorial have not changed too significantly since the tutorial was created.

OVERVIEW OF THE DISSERTATION

This dissertation presents research into resolving the problems that arise when reusing World Wide Web documents in tutorials. This research used an experimental system called the Wtutor system as the testbed for the research effort. Wtutor is a system that allows a tutorial author to create tutorials as trails through WWW documents. Chapter 2 reviews the research that has been done in the area of providing trails, paths, and guided tours through hypertext materials. Chapter 3 describes the basic tutorial support that provides solutions to the filtering and navigation problems. Chapter 4 presents a set of prefetching techniques and policies that can resolve the retrieval problem. Chapter 5 describes an approach to solving the validation problem. Finally, Chapter 6 describes some of the possible areas of future research.

CHAPTER 2: RELATED WORK

The filtering and navigation problems have long been recognized in the hypertext research community, and much effort has gone into the exploration of solutions for these problems. The article which is most often credited with inspiring the concept of hypertext, Vannevar Bush's "As We May Think", was an attempt to deal with the filtering and navigation problems in the context of a large indexed database of information [Bush].

Bush described a conceptual machine for creating "trails of thought". His *memex* was a machine containing massive amounts of information on microfilm, with tools for indexing and searching the data. He envisioned trail blazers—researchers who would find connections between elements in the memex data store and create sequential trails linking these elements into a conceptual whole. These trails of thought could then be reviewed by the creator and shared with others.

Bush's concept of linked materials is generally credited with inspiring the development of hypertext. However, the link plays a much more prominent part in hypertext systems than Bush envisioned. Rather than databases of indexed information with occasional trails, hypertext has evolved as sets of highly interconnected nodes of information, with multiple links both into and out of the nodes. Unfortunately, the proliferation of nodes and links leads to the filtering and navigation problems.

THE FILTERING AND NAVIGATION PROBLEMS

In a nutshell, the filtering and navigation problems boil down to too many choices. A reader at a typical hypertext node is presented with several links to choose from, and somehow the reader must decide which one to select. In some cases, the best choice is not even one of the links that is displayed on the document that the reader is viewing. Some of the symptoms of the filtering and navigation problems have been described as the "embedded digression" problem, where a reader follows links and ends up sidetracked further and further from the relevant data, and the "art museum phenomenon", where the range of data presented to the reader is too flat, rich and endless for the reader to make sense of what is important and how it all fits together [Dowling].

Some hypertext researchers have claimed that the navigation problem is not really a problem [Landow]. The claim is that "good writing" is the solution; in other words, the navigation problem is merely an artifact of poorly designed and written hypertext documents. However, this claim is generally made in the context of self-contained hypertexts; that is, documents where a single author or team of authors create the entire document. Reuse of existing documents does not allow the "good writing" solution, because the tutorial author is not necessarily the owner of the source material and may therefore not have control over the contents of source material.

This was illustrated in the sample tutorial (described in Chapter 1) in the first few documents of the tutorial, where the tutorial author wanted the student to proceed from the biography document to the chronology document. Because the biography document is

owned by the Martin Luther King, Jr. Papers Project the tutorial author is not able to add a link to that document connecting to the chronology document.

Even if the tutorial author could add a link to the biography document, doing so may not be desireable. Such a link would serve a specific purpose for the tutorial, but may not be appropriate for other uses of the biography document. What if another tutorial author wanted to include the biography document in a different tutorial? Would the biography document include two "Next" links? Allowing the reuse of documents implies that sequencing information should not be embedded in the documents.

TRAILS, TOURS AND PATHS

To solve the filtering and navigation problems, researchers returned to Bush's original concept of a trail of thought. The intent is to provide a reader with a sequential set of nodes to visit, thereby reducing the cognitive load on the reader by eliminating the filtering and navigation decisions. In the TEXTNET system, Trigg introduced the path mechanism, which provided a sequential history of the nodes a reader visited during a session and allowed the reader to back up and retrace the path [Trigg86]. This feature is now nearly universally provided in all hypertext systems. Trigg expanded on the idea of the path in the Notecards system with Guided Tours. A Guided Tour was a sequential set of stops called Tabletops, where a Tabletop was a set of Notecards which was displayed together. The Guided Tour interface allowed readers to enter or leave a tour, sequentially visit the stops on the tour, or jump between stops on the tour [Trigg88].

Zellweger provided a path mechanism, called a script, in the Cedar programming environment [Zellweger88]. The original version of scripts merely allowed for the sequential playback of documents; Zellweger expanded on this idea with Scripted Documents, which introduced conditional and programmable paths [Zellweger89]. With Scripted Documents readers could single-step through a script, allow the system to step through the script automatically, or browse a script by selecting specific entries from the script to visit.

Hammond and Allinson also used the guided tour approach to implement a tutorial mechanism [Allinson89]. This fit within their overall metaphor of navigation as a travel holiday. The guided tour allowed the reader to visit a sequence of frames. The reader could "step off the bus" at any point of departure; features were provided to enable the reader to rejoin the tour at the point of departure or to return to the start.

In the Grolier project, readers could select guides to lead them through a set of historical materials in a multimedia database [Laurel]. The guides represented a point of view; the set of links from each node were ranked and subsetted by how well they matched the topics which characterized the guide.

One rather extreme approach to eliminating the navigation problem was adopted in the Book Emulator, which presented materials as if in a physical book [Benest]. Not only did the Book Emulator linearize the materials, it presented the materials as pages in a book. When the reader requested the next page, the system would visually depict the

turning of a page. This system allowed links within a book; it does not appear to support external links, which limits its usefulness.

In an effort to eliminate the work involved in creating and maintaining Guided Tours, Guinan and Smeaton developed a system for dynamically generating Guided Tours of retrieved material [Guinan]. Their system would use information retrieval techniques to retrieve hypertext nodes relevant to a reader's query, then use structural information from the links within the nodes to order the nodes, rather than using the traditional information retrieval technique of presenting nodes in order of how well they matched the query. One drawback of this approach is that it requires typed links that provide structural information about the relationships between the selected nodes. The vast majority of links in the World Wide Web are untyped, and therefore do not provide the structural clues required by this approach.

One case where paths have been used to good effect is in Perseus [Marchionini]. Perseus is a large hypertext database of multimedia materials regarding ancient Greece. A path mechanism is provided in Perseus, where paths can be created by both instructors and students.

A primary reason that the use of trails, tours, and guides did not become more widespread was that most research on these techniques was done with closed, self-contained hypertext systems, such that each system had a limited number of nodes to choose from. As noted in [Landow], there simply wasn't a large enough base of widely accessible material to choose from. The explosion in popularity of the World Wide Web

has now provided a huge base of accessible material. The proliferation of nodes and links on the Web has revived interest in trails as an important tool for use with hypertexts.

TOURS ON THE WORLD WIDE WEB

Recent researchers have created path mechanisms for the Web. The BRIO system is one system that has a path mechanism [Roscheisen]. The primary purpose of BRIO is to enable the creation and retrieval of annotations on Web documents. These annotations are stored separately from the annotated document. When annotated documents are retrieved, using a modified Mosaic browser, the existence of the annotations is indicated by icons displayed within the document. Selecting the annotation icons causes the annotations to be retrieved and expanded within the document display.

In BRIO, tours are implemented as a list of links to annotations. A reader takes a tour using a two-pane browser. In one pane the browser shows the list of links that make up the tour; in the other pane the annotated Web documents are displayed.

This approach requires the use of a specific browser; that is, taking a tour requires the use of the BRIO browser. It also requires an annotation on any document that is to be included in the tour. The annotation feature allows a built-in mechanism for the tour author to provide commentary on the included documents, so that rather than having to create introductory or transitional Web documents, the author could include such material in the annotations to the Web documents that are incorporated into the tour.

To avoid the dependence on a particular browser other researchers have built systems that use proxy servers and CGI scripts to implement tours. The reader may use

any standard Web browser to take a tour. Walden's Paths is one such system [Shipman]. Walden's Paths implements paths via a path server, which acts as an intermediary between the reader's browser and the servers that provide the documents in the path. The path server stores the path definitions. Each path is defined as a linear set of URLs; the path may also include annotations for each URL in the path.

When a reader initially connects to the path server, the server presents a list of paths that are defined to the server. When the reader selects a path, the server initiates the tour. As the reader steps through the tour, the path server performs several functions. The path server determines which URL the reader should see. That URL is retrieved by the server. The server then modifies the content of the URL to include navigational controls and information as well as any annotations specified in the path. A sample page from one of the Walden's Paths paths is shown in Figure 7. The left and right arrows at the top of the Web page in Figure 7, the Walden's Paths logo, the numbers with arrows, and the initial text are all inserted by the path server. The arrows allow the reader to move back and forth along the path. The logo takes the reader back to the list of available paths for the server. The numbers graphically represent the path, including the reader's current position in the path, and allow the reader to move to points along the path. The text that immediately follows the numbers comes from the annotations in the path.

In order to track the reader's progress through the path, the path server modifies each link in the Web page so that selecting the link will activate the CGI scripts that

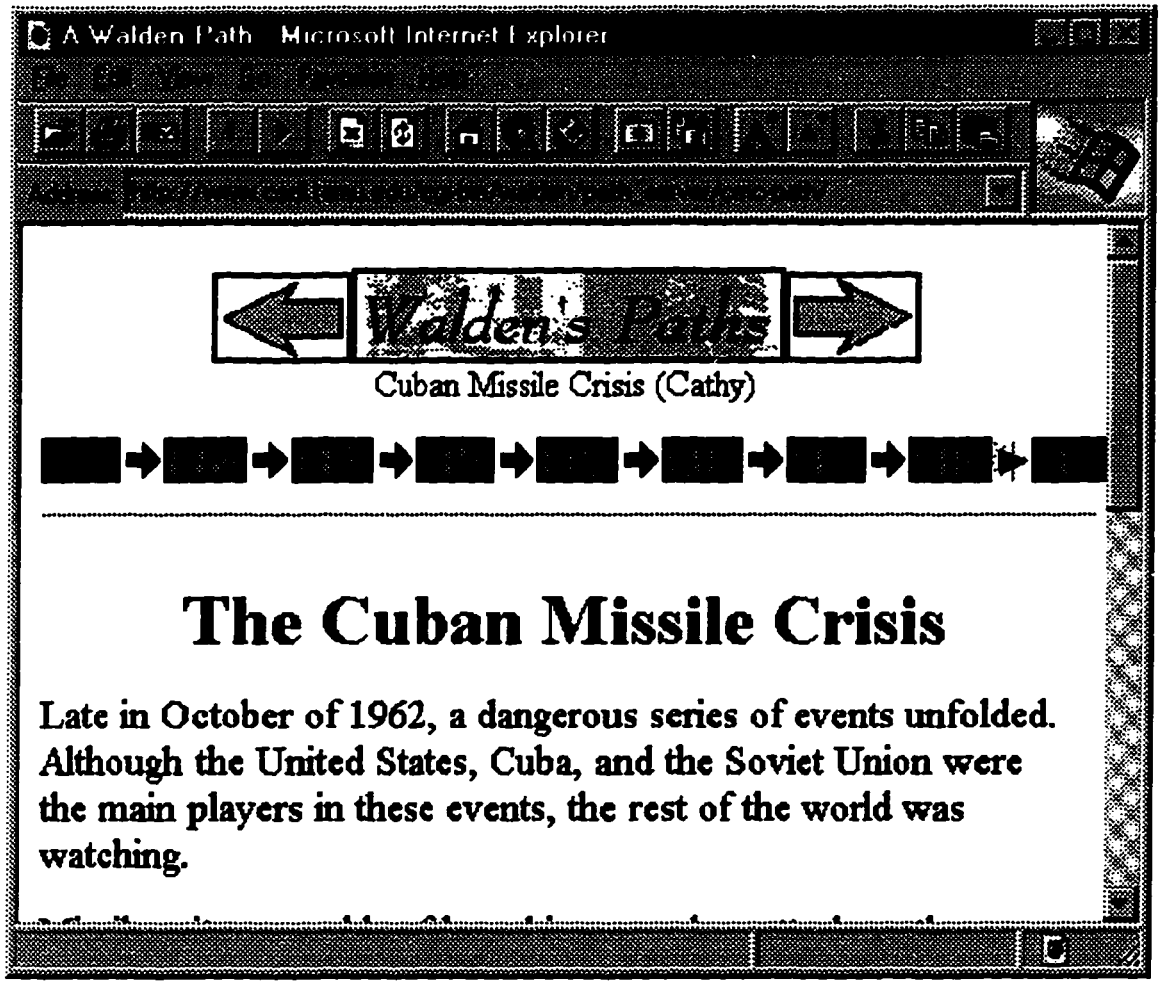


Figure 7 Sample page from Walden's Paths path server

implement the path mechanism. If the reader selects a link that goes off the path, the path server inserts the Walden's Paths logo and a link to take the reader back to the path.

Because the path server acts as an intermediary between the reader's browser and the servers that provide the documents in the path, this approach requires twice the normal network traffic because all but inlined objects are retrieved in a two-stage process. One way in which Walden's Paths mitigates this problem is by caching retrieved documents, thereby minimizing the first-stage retrievals. Another side-effect of this approach is that

the embedded documents are not necessarily shown as the reader would see them if the reader directly retrieved the document. Most browsers display the target URL when the reader places the mouse over a link; because the links are modified for Walden's Paths, the reader would see the invocation of the CGI script rather than the target URL. Walden's Paths also allows the path author to specify formatting changes to the included documents; although this may improve the suitability of the documents for purposes of the path, it is questionable whether it is appropriate to modify a document while also displaying the URL that identifies it at the same time.

The Footsteps system works in much the same way as Walden's Paths [Nicol]. In the Footsteps system, a tour is defined in a file which contains a sequential list of URLs to be visited in the tour. The tour is conducted via a CGI script which acts as an intermediary between the reader and the materials within the tour. Whenever a document is retrieved, the script modifies any links within the document and adds navigational controls before displaying materials to the reader.

Footsteps differs from Walden's Paths in some of the details of implementation. Where Walden's Paths displays a graphical representation of the path on each page, Footsteps provides a link to an index. The index provides a complete list of the entries on the tour, with the titles of each entry taken from the source documents. Footsteps does not provide a means for the tour author to modify the display of the included documents or to include any annotations. As with Walden's Paths, Footsteps does require that retrieval occur in two stages.

With trails and tours becoming viable approaches to resolving the filtering and navigation problems on the World Wide Web and allowing the reuse of existing materials, it becomes imperative to address the retrieval and validation problems.

CHAPTER 3: BASIC TUTORIAL SUPPORT

A World Wide Web based tutorial system that allows the reuse of existing materials must provide some means of leading students through materials; i.e. a path or guided tour mechanism. The tutorial author must be able to create paths. The system must be able to track the student's progress along paths and use that information to guide the student. In order to explore the issues and possible solutions to the problems that arise in Web-based tutorials, an environment was developed that supports the creation and use of Web-based tutorials.

The Wtutor environment is composed of four major components: a tutorial map, Wtutor, Tmosaic, and the World Wide Web. The Wtutor environment is pictured in Figure 8. The *tutorial map* is a construct that identifies the contents of the tutorial and the ordering of those contents. Wtutor is an Allegro Common-Lisp program that tracks the student's progress through the tutorial and provides guidance to the student. Tmosaic is a modified version of NCSA Mosaic that is used to retrieve and display the documents that make up the tutorial.

TUTORIAL MAPS

Each tutorial is defined in a tutorial map. The tutorial map contains the URLs for each of the documents within the tutorial. It also specifies the ordering of those URLs via precedence relations between the URLs. Logically, the tutorial map can be thought of as a

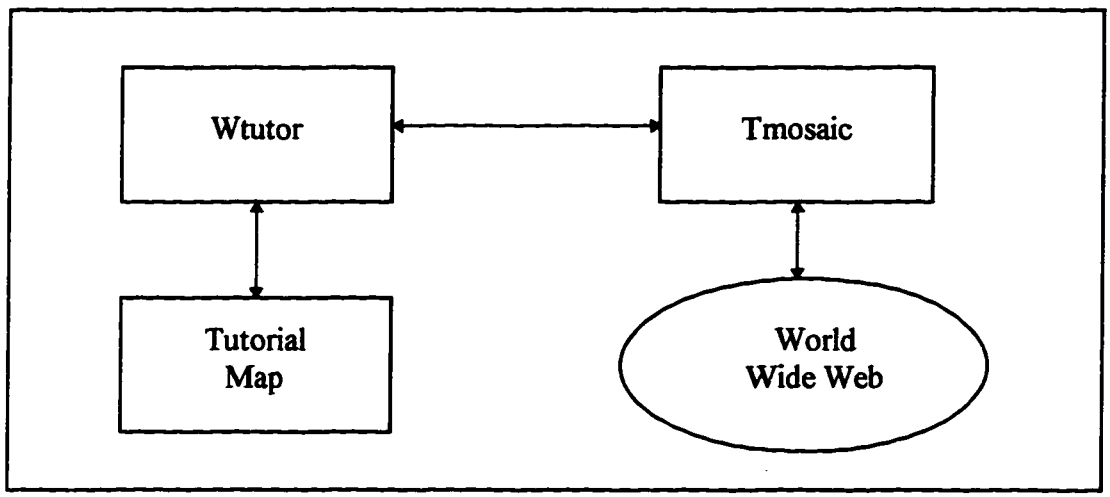


Figure 8 Wtutor environment

directed graph, where the nodes of the graph are URLs and the edges of the graph are precedence relationships between the nodes. See Figure 9, which shows a graphic representation of a tutorial map with four documents. In this example, the Introduction document precedes the Biography document, which precedes the Chronology document, which precedes the Civil Rights document. Physically, the tutorial map is stored as a text file in Lisp syntax.

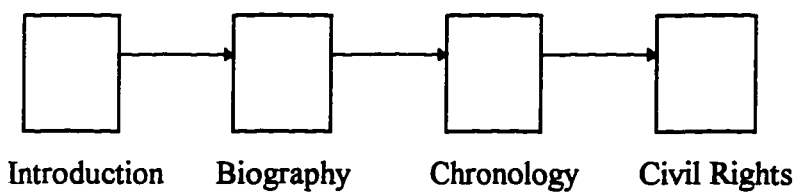


Figure 9 Sample tutorial map

The precedence requirements in the tutorial map indicate the logical ordering of the documents that make up the tutorial. In Figure 9, the precedence relationship between

the Introduction and Biography documents indicates that the Introduction document should be visited before the Biography document is visited by the student.

The tutorial map allows the author to specify that documents be displayed together; for example, a graphic image such as a map could be displayed along with a commentary that describes a historical trade route that can be found on the map. This feature is similar to tabletops in Notecards [Trigg88] and the multi-headed/multi-tailed links in Multi-head/Multi-tail Mosaic [Ladd]. The together relationship can be used to trigger playback of audio documents that comment on displayed documents.

Tutorial maps allow the definition of groups of URLs. This may make it easier for an author to logically organize the tutorial. For example, in Figure 9, the tutorial author could group together the Introduction, Biography and Chronology documents into a group called "Intro". The author can specify precedences between groups, or between a group and a URL. The group feature is one of the reasons that an author does not have to specify a completely linear ordering of the materials. The author may identify two groups of URLs and specify that one group precedes the other, without specifying an ordering of the URLs within the groups. Wtutor has selection rules built in for determining which URLs to visit.

The author can also identify alternative relationships between URLs or groups; that is, if URL A and URL B are alternatives, the student does not need to view B if the student has already viewed A. For example, if the Biography document in Figure 9 were provided by an undependable server, the tutorial author may want to create an alternative

document that would provide similar content to the Biography document, so that when the Biography document was unavailable the student could proceed with the tutorial. The alternative relationship, combined with the group feature, allows the author to specify alternative paths within a tutorial. An alternative path may be selected because of the student's choice or because of availability of materials at the time the student is taking the tutorial. Most guided tour implementations do not allow alternative paths through the materials.

TMOSAIC

Tmosaic is a modified version of NCSA Mosaic. It is used to retrieve the Web documents that make up the tutorial and display those documents to the student or tutorial author. The primary modification to NCSA Mosaic was the addition of a socket and corresponding message protocol so that Wtutor could communicate with Tmosaic. The message protocol allows Tmosaic to notify Wtutor of events such as retrieval of a URL, submission of a form, or student requests such as "Next URL". In turn, Wtutor can send requests to Tmosaic such as "Load URL" or "Display Msg".

Another addition to Tmosaic is a menu for selection of tutorial-related functions. Figure 10 shows the Tutor menu. The "Next" and "Previous" options are used to notify Wtutor that the student wants to move forward or backward within the tutorial. The "Load Tutorial Map" option displays a popup window as shown in Figure 11. The student or the tutorial author uses this window to specify a tutorial map to be loaded by Wtutor.

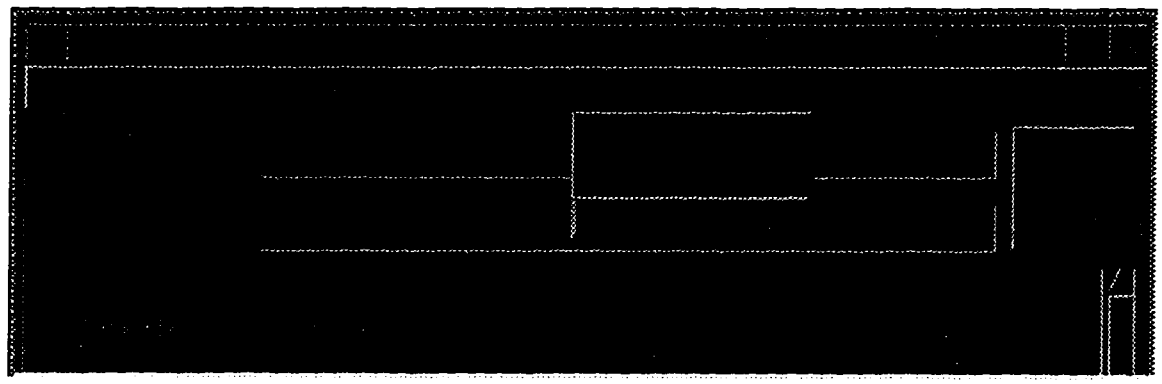


Figure 10 Tmosaic Tutor menu

Whenever a URL is retrieved, Tmosaic sends messages to Wtutor specifying that the URL has been retrieved, and asking whether any of the links contained within the URL should be highlighted. Wtutor specifies that a link to a URL should be highlighted when the URL is included in the tutorial map and all precedence requirements for that URL have been met. The tutorial author can specify the highlight color to be used.

As the student proceeds through the tutorial, the highlighted link displays will indicate good next choices to visit. If the student does not have a highlighted link on the currently displayed link, the student can have Wtutor select the next URL to visit by selecting the "Next" option under the "Tutor" menu.

WTUTOR

Wtutor tracks the student's progress through the tutorial map. Wtutor reads in the tutorial map when a student requests the Load Tutorial Map option. When the student is opening the tutorial for the first time, Wtutor finds the first document of the tutorial and instructs Tmosaic to retrieve that document. When Wtutor receives the notification from

Tmosaic that a URL has been retrieved, Wtutor marks the URL entry in the map as visited. If the URL is part of a group, Wtutor also checks to see if all the URLs within the group have been visited; if they have, the group is marked as visited. If the URL has an alternative, the alternative is marked as not needing to be visited.

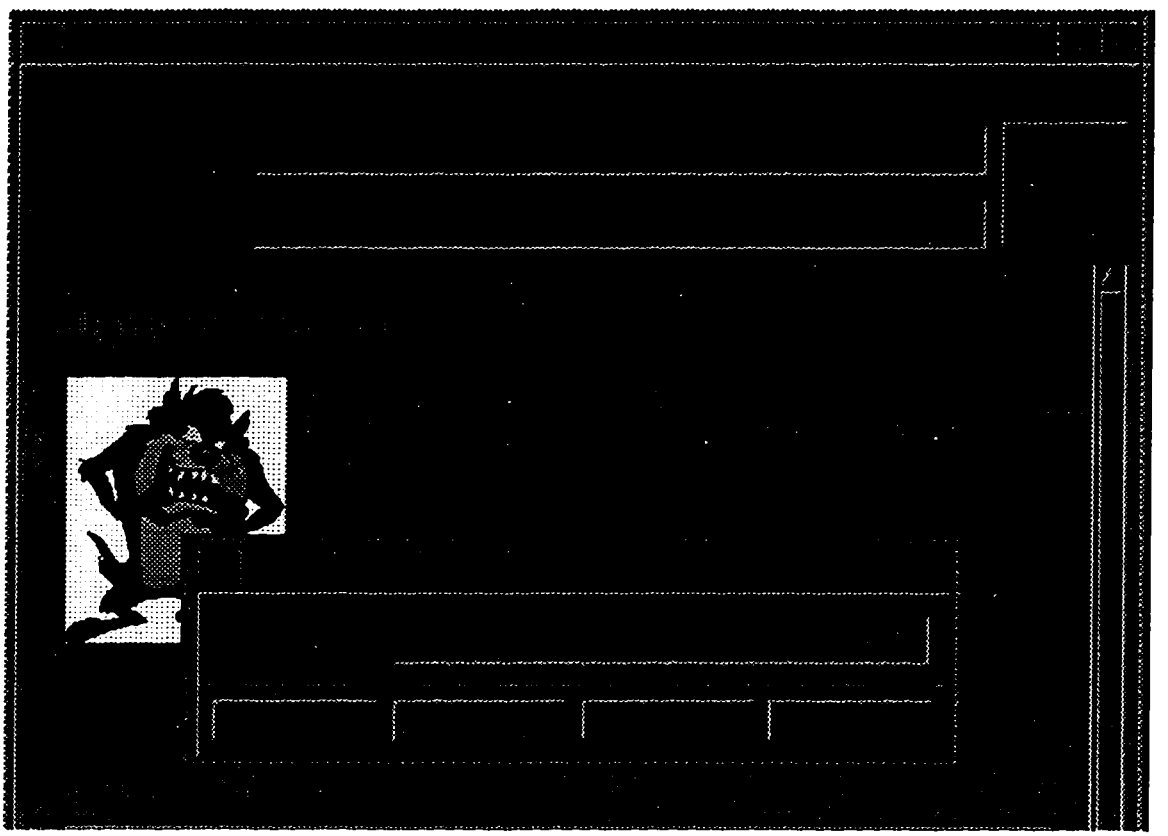


Figure 11 Tmosaic Load Tutorial Map popup window

When Wtutor receives notification that a URL has been retrieved, it checks the tutorial map to see if the retrieved URL has a together relationship with another URL. If there is a together relationship, Wtutor requests that Tmosiac retrieve the other document. If the other document needs to be displayed in a Tmosaic window, Wtutor specifies that a

new window should be opened for display of the other document, so that both documents can be viewed at the same time. If the other document is viewed with an external viewer, such as xv, Wtutor does not request a new window.

When the student requests the next URL, Wtutor finds a URL that is ready to visit. Wtutor follows a set of rules in searching for the next URL. The overriding rule is that all the precedents of a URL must have been visited before the URL is selected as the next URL. Wtutor looks first for a consequent of the current URL; if there are none that are acceptable then consequents of the previous URL visited are checked. If none of these are acceptable, Wtutor looks for URLs within the same group as the current node. Finally, Wtutor looks for any URL within the map whose precedence requirements have all been met. If all URLs within the tutorial map have been visited, the student is notified that the tutorial has been completed.

The presence of Wtutor and the use of the tutorial map help eliminate the filtering and navigation problems for the student. The student is free to follow any links that are displayed in the tutorial documents, so exploratory learning is not restricted. If the student follows links to documents that are not part of the tutorial, the Tutor menu provides a means for the student to return to the tutorial. The tutorial map defines the domain of the tutorial, so the student does not have to filter out any tangential links. Because Wtutor and Tmosaic either highlight links or allow the student to request the next document the student does not have to deal with the navigation problem either.

AUTHORING TUTORIALS

In the Wtutor environment, the key responsibility of the tutorial author is to create the tutorial map. While the tutorial map eliminates the filtering problem for the student, it does not solve the problem for the tutorial author. The author must still find or create the documents to include in the tutorial. For this task the author can use tools such as the MetaCrawler [Selberg] to find URLs that may be appropriate. The author must evaluate the candidate documents to determine fitness for inclusion in the tutorial.

The tutorial map is created online by the author. The tutorial author can add or delete URLs from the tutorial map using Wtutor. The author can either name a fully qualified URL to be added to the map, or retrieve a URL using Tmosaic and then specify that the retrieved URL should be added to the tutorial map. The author also specifies the precedence relations in the tutorial map.

Wtutor provides several identifiers to ease the tasks of adding URLs and precedence relations to the map. The most recently retrieved URL and the previously retrieved URL are tracked and can be referred to by keywords *curl* (for current *URL*) and *previous*, respectively. In addition, the tutorial author can mark a particular URL and then refer to the mark later, as in adding a precedence relation between the current URL and the marked URL.

TOOLS FOR GENERAL TRANSITION PROBLEMS IN WWW BASED TUTORIALS

Creating a good tutorial requires establishing a "flow" through the materials of the tutorial. When an author is creating all the contents of a Web-based tutorial, the author

has complete control over the contents. This control means the author can build a consistent look and feel to the materials and make sure that the transition from each Web page to the next page is smooth and logical.

Reusing documents from the Web makes it more difficult to establish a smooth flow between the elements of the tutorial. The tutorial author no longer has control over the look and feel of each Web page, other than to reject those that are clearly unsuited to the tutorial. One problem that can occur is inconsistent linkage between Web pages. For example, the tutorial author may include a sequence of pages from one author, where those pages each have a "Next" link that the student can follow; while another set of pages may have a "Continue" link.

A more subtle problem concerns the transition between URLs when there is no link between those URLs. When a student is viewing one URL and follows a link to another URL, the transition from the first to the second is aided by the context and content of the link display in the first URL. The information provided by the context and content of the link display allow the student to make inferences regarding the relationship between the two URLs. The student loses this information when going from one URL to another without a connecting link, e.g. via the "Next URL" feature of Wtutor.

One way of avoiding the transition problems is for the author to create introductory or transitional materials and insert these materials between URLs that are logically but not physically linked. Another way is for the author to use the together relationship, which specifies URLs that should be viewed together. This allows the tutorial

author to create explanatory material that can be displayed together with URLs that the tutorial author did not originate. The together URLs could contain audio segments, graphics, outlines, or any other material created by the tutorial author to help explain why the student is viewing the selected material and to help put the material in the proper context.

STUDENT FEEDBACK

Tutorials frequently include self-assessment sections that help students determine how well they have learned the material in the tutorial. This type of feedback helps reinforce the materials for the students. Wtutor provides a means of including assessment points in tutorials.

In addition to URLs, a tutorial author can include assessment points in the tutorial map. Wtutor supports author specified multiple choice items as the assessment points. The author creates assessment points using Tmosaic and a fill-out form. This form is shown in Figure 12. As shown in Figure 12, the form has entries for a title for the assessment item, the question, a set of up to five possible answers along with an indication of whether the answer is correct, and an optional remedial URL (not visible in the figure). The author fills in the form and submits it; Wtutor captures the information from the form and creates an assessment point in the tutorial map. The author can add precedence requirements between assessment points and URLs or other assessment points. Assessment points can also be included in groups or in together or alternative relations.

When the student comes to an assessment point in the tutorial, Wtutor generates an HTML form containing the question and possible answers. Wtutor then directs Tmosaic to load and display the generated form. Figure 13 shows the form generated from

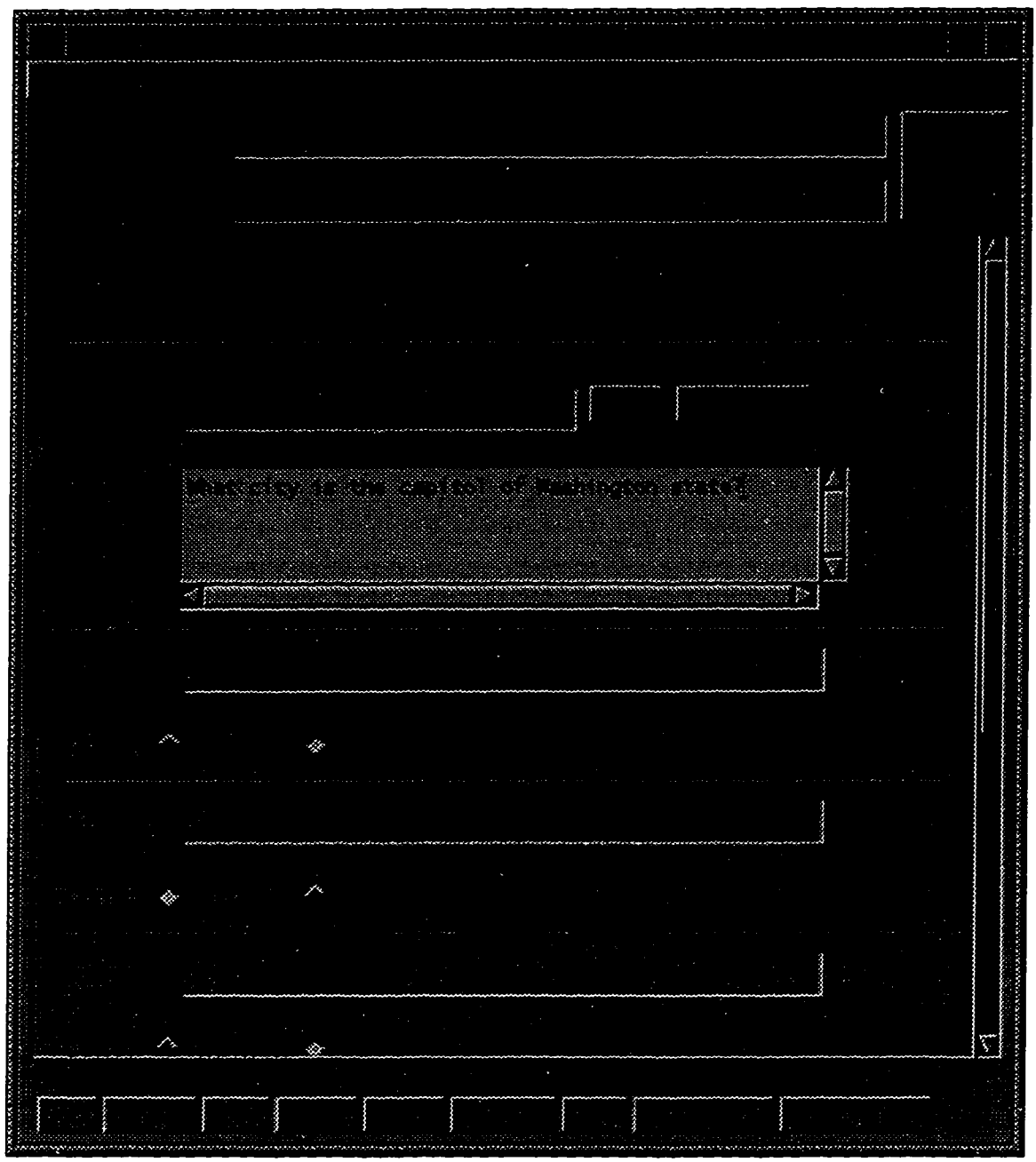


Figure 12 Fill-out form for creating assessment items

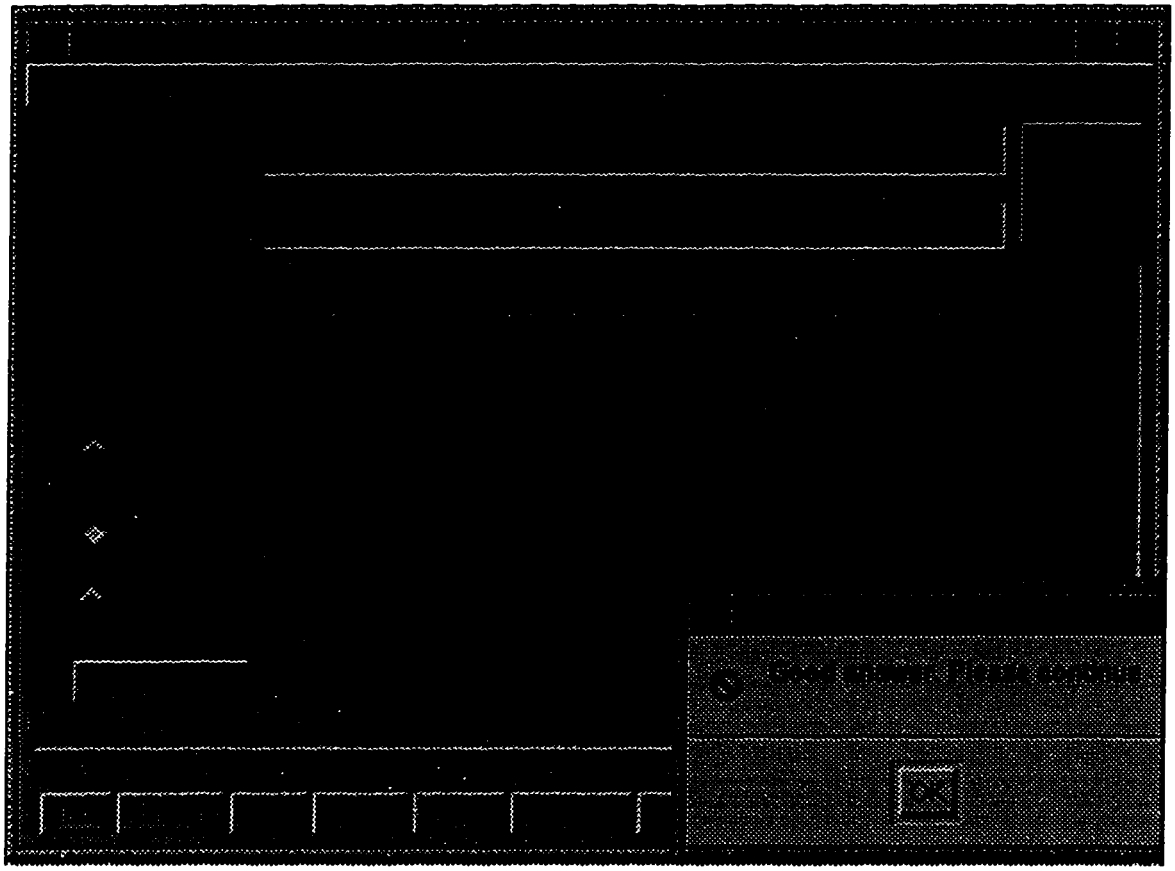


Figure 13 Sample assessment item with Wtutor feedback

the assessment item illustrated in Figure 12. The student selects an answer from the generated form and submits the form for evaluation. Tmosaic passes the student's response to Wtutor, which compares the student's response to the correct response. If the response is correct the assessment point is marked as completed. Wtutor then notifies the student whether the response was correct and either directs the student to the next URL or assessment point or to the remedial material. Figure 13 also shows a sample of Wtutor's feedback to the student's response to an assessment item. Because the

assessment process is implemented via forms and Wtutor, the tutorial author does not have to write any HTML for the assessment points.

The Wtutor environment as described above provides the basic support for creating and reading tutorials. While the combination of Wtutor and the tutorial map resolve the filtering and navigation problems, the retrieval and validation problems must still be addressed. These problems are further explored in the following chapters—in Chapter 4 the retrieval problem and a possible solution are presented; in Chapter 5 the validation problem and a possible solution are addressed.

CHAPTER 4: SOLVING THE RETRIEVAL PROBLEM

The distributed nature of the World Wide Web introduces another potential problem in the reuse of materials. That problem is access time—the time it takes to retrieve a document. From the reader's point of view, access time is measured from the time when the reader requests a document to the time the document is displayed for the reader. Therefore, access time includes the time it takes to register the request at the Web server, the server processing time, transmittal time as the data is sent across the network, and the time the browser takes to format the document for display and display it on the screen.

The retrieval of documents takes time under the best of circumstances; the retrieval time can be extended due to heavy network traffic or load on the Web server. Delays due to retrieval time can be a serious impediment to the student's learning experience. The student's intended train of thought can be interrupted, and the student can get bored, distracted or frustrated. The variability of retrieval times on the WWW is often especially frustrating.

Most delays due to retrieval time can be eliminated by a tutorial system that prefetches documents prior to the student's request for the documents. Prefetched documents can be displayed to the student almost instantaneously, so the student does not experience any delay between the request for a document and the display of that document.

The tutorial system's prefetching policy was designed to satisfy specific goals. The goals which have been established for the tutorial system's prefetching policy, ranked in order of precedence, are as follows:

1. Each URL within the tutorial should be prefetched by the time the student requests the URL. In other words, as long as the student is following the tutorial map, the system should retrieve documents before the student requests them.
2. Minimize network resource consumption. Responsible use of the Internet requires that unnecessary traffic be avoided as much as possible.
3. Minimize unnecessary cache consumption. The amount of RAM or disk available for caching prefetched documents may be limited; if so, excessive prefetching may cause cache overflows that could mean documents are no longer available when the student requests them. This would mean the document would have to be retrieved again, so both the first and second goals would be violated.
4. Prefetching should be both interruptable and resumable. A student's request should not be delayed by the completion of a prefetch (unless the prefetch is of the document the student is requesting, in which case completion of the prefetch would minimize the delay). If a prefetch is interrupted, resumability would prevent the loss of any data retrieved prior to the interruption.

If the first goal were the only goal, then the prefetching policy would be simple: retrieve all URLs within the tutorial map as soon as the map is loaded. Assuming enough cache space were available, this would ensure that all documents would be prefetched. However, this approach could consume quite a bit of unnecessary bandwidth and server time, if the tutorial had multiple alternative branches or if the student quit before finishing the tutorial.

The Browser Buddy product from Softbots, Inc. is a Web browser which uses the "prefetch all at initiation" approach [Softbots]. Users use the browser to create lists of URLs. At a later time, a list can be requested. When a list is requested, the Browser Buddy prefetches all the URLs in the list and the user can then browse the URLs. This policy has also been used in Walden's Paths, although the prefetching is only done at the

path server, which means that the documents must still be transmitted from the server to the browser when the reader requests the next document [Shipman].

A better prefetch strategy would take into account the student's position within the sequence of URLs that are to be viewed. A dynamic prefetching strategy is implemented within the SETS system [Steere]. With the SETS Web browser, a document can be treated as a dynamic set of links. When a document has been retrieved and displayed to the user, the user can initiate prefetching of the dynamic set. The browser then initiates multithreaded prefetching of all the URLs within the document. The SETS browser has two options for "throttling" prefetching. The user can (1) specify a limit on the number of outstanding prefetch requests or (2) limit how many URLs to prefetch, e.g. if a dynamic set had ten links in it and the limit were three, the first three URLs would be prefetched initially, but the fourth would not be prefetched until the user actually looked at the first prefetched URL. This approach is primarily geared toward usage with search engines, where a sequential list of URLs that match some search criteria is returned to the user. This list is then treated as a dynamic set.

The other primary approach to prefetching in the WWW has been server-based. The fundamental concept for server-based prefetching is that servers can track accesses to URLs and use historical patterns to predict client behavior. Both [Touch] and [Padmanabhan] describe systems where servers track requests for URLs from clients to identify frequent patterns of access. When a client requests a URL, the server returns not

only the requested URL, but a list of potential next URLs with associated probability ratings that indicate how likely it is that the client will request the URLs.

The client can then select which of the URLs on the list to retrieve. The client may decide not to prefetch some of the URLs for a number of reasons. The client may already have some of the URLs cached; it may have limits on how many URLs it will prefetch; or it may require a minimum probability of retrieval that some URLs on the list do not meet. [Bestavros] describes a slightly different approach. Server predictions are made in the same way, based on historical patterns. However, when the server sends the requested URL, it also automatically sends the predicted URLs.

The server-based approaches to prefetching require servers that will provide predictions to the client. If the servers do not make predictions, then the client has no way of knowing what to prefetch. The server based approach also fails to provide any assistance to the client when a reader goes from one server to another. Server-based prefetching is not well matched to tutorials that include materials from a number of servers.

CONCEPTUAL MODEL OF PREFETCHING

We start with a simplified model of the process of reading through a tutorial. This model initially assumes a number of restrictions for purposes of clarity. These restrictions will gradually be relaxed in order to more accurately model tutorials on the Web.

In the simplest version of the model, the student reads through a tutorial without the system doing any prefetching of documents. In this model, the student requests the

first document, the system retrieves it, the student reads the first document and then requests the next. The process repeats until the student has completed the tutorial. "Reading" a document may mean actually listening to, viewing, or executing the document. The time interval between the presentation of the document to the student and the student's request for the next document is the *visitation time* for the document. The time interval between the student's request for a document and the display of that document to the student is the *retrieval time* for the document. The process of reading through a tutorial can be represented graphically as in Figure 14.

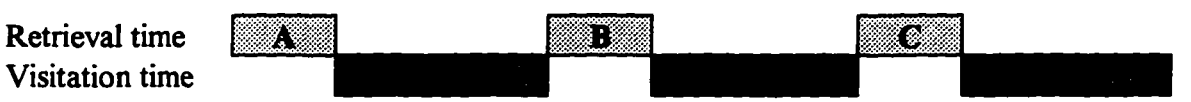


Figure 14 Reading tutorial without prefetching

A prefetching policy that perfectly met all the goals described earlier would eliminate the gaps between the visitation times while still only retrieving the documents actually visited by the student. Figure 15 graphically represents the process of reading through a tutorial when a perfect prefetching policy is in place.

Figure 15 Reading tutorial with perfect prefetching

A perfect prefetching policy is possible under restricted circumstances. The required restrictions are (1) the retrieval and visitation times are known for all nodes, (2) the sequence of visitation of nodes is known in advance, and (3) the retrieval time of each node is less than the visitation time of the preceding node. Under these restrictions, a

tutorial system can implement "just in time" prefetching. This would retrieve most URLs in advance of the student's needs and limit the documents retrieved to a minimal set. Upon display of a document to the student, the system can then calculate the *latest prefetch initiation time* (LPIT) for the next document, which is the latest point in time that a prefetch can be initiated and still complete the prefetch before the student requests the document. The latest prefetch initiation time can be calculated by subtracting the retrieval time for the next document from the visitation time for the current document.

Because the retrieval times are restricted to being less than the visitation time of the preceding nodes, and the visitation and retrieval times are known, this policy is guaranteed to prefetch all documents before the student requests them. Because the policy delays prefetch initiation until the latest possible time, the minimum number of documents are retrieved. In the worst case, one extra document would be retrieved; this extra retrieval would only occur if the student ended the session without completing the tutorial. Since the minimum number of documents are retrieved, cache and network utilization is minimized.

The assumption that all retrieval times are less than the preceding visitation time is key in that it ensures that the prefetch of each URL can be completed while the preceding URL is being read by the student. However, this restriction is not always a realistic one. For example, a short text document could be followed by a document containing several imbedded graphic images. In such a case it may only take 15-20 seconds to read the text document while requiring 45-50 seconds to retrieve the following document. Such an

example is illustrated in Figure 16, where the retrieval time of URL C is longer than the visitation time of URL B. If the prefetch for document C is not initiated until document B is displayed, the prefetch will not be complete when the student requests document C.

Figure 16 Reading tutorial with retrieval times greater than visitation times

Relaxing the restriction on retrieval times means that the latest prefetch initiation times must be calculated at the beginning of the tutorial because some documents may need to be fetched before their preceding document is displayed. In the example in Figure 16, the prefetch of document C would need to be initiated before document B is displayed, while the student is still reading document A. However, this leads to another potential problem. In the example, the prefetch of C would overlap with the prefetch of B. If prefetching is implemented as multithreading, this may not be a problem, although the retrieval times may be longer under multithreaded prefetching.

If prefetching is not multithreaded, the prefetch initiation times must be adjusted to avoid overlapping. In the example in Figure 16, the prefetch of document B would need to be initiated early enough that the prefetch would complete before the required prefetch initiation time for C. In general, the LPITs would be calculated in a two step process.

First, we define the *expected arrival time* for each node as the point in time when we expect the student to request display of that node. This is the time at which the student is expected to arrive at that node in the tutorial. Assuming that all nodes are prefetched

prior to being requested, the expected arrival time for each node is simply the sum of the visitation times for all the preceding nodes in the visitation sequence. Because it is known in advance which nodes will be visited in what sequence, the LPITs can be calculated by starting at the end of the visitation sequence and working backward. For each node, the LPIT is calculated by subtracting the retrieval time for that node from the lesser of (a) the expected arrival time of the node or (b) the LPIT of the following node. Using this approach, the example in Figure 16 is transformed as in Figure 17, where again the only student-experienced delay is at initiation of the tutorial.

Figure 17 Adjusting for retrieval times greater than visitation times

With the removal of the restriction that retrieval times be less than visitation times, we can no longer guarantee that all documents will be retrieved before the student requests them. In the extreme case, if the retrieval time for each document is greater than the visitation time of the preceding document, the student will experience delay between each node. However, wherever possible the retrievals will be overlapped with visitations, so student experienced delays will be minimized, and no documents will be retrieved until necessary, so network utilization will also be minimized.

The implementations of guided tours on the World Wide Web generally restrict the tours to a linear sequence of nodes. Therefore, the restriction that the sequence of visitation of nodes be known in advance is not an unreasonable one. However, the tutorial

map approach allows for alternative paths, nodes that are displayed together, and points where the student may select which node to visit next. A prefetching policy for the tutorial map model must therefore make allowances for not always knowing in advance the visitation sequence.

Dealing with nodes that are displayed together is not difficult, because the together nodes simply present a special case of overlapping prefetches. The policy would have to adjust the LPIT for one of the together nodes to ensure that retrieval completes before the LPIT of the other node; this solution generalizes for multiple together nodes.

Alternative paths and points where the student may select which node to visit next present more complexity, because these features mean that the system cannot determine in advance the visitation sequence. The prefetching policy must allow for all likely paths through the tutorial, and then adjust as the student progresses through the tutorial. For example, Figure 18 shows a tutorial map with alternative nodes C and D. In the figure, the visitation times are shown above the node, retrieval times below the node. This example illustrates several of the complications that arrive with alternative paths.

First, the system does not know in advance whether the student will visit C or D. This means that the prefetch policy must provide for both C and D to be prefetched by the time the student finishes visiting node B; this also means the prefetch policy must avoid overlapping prefetches for C and D. Second, the prefetch policy must recognize that node E may be visited after either C or D has been visited. This complicates the calculation of

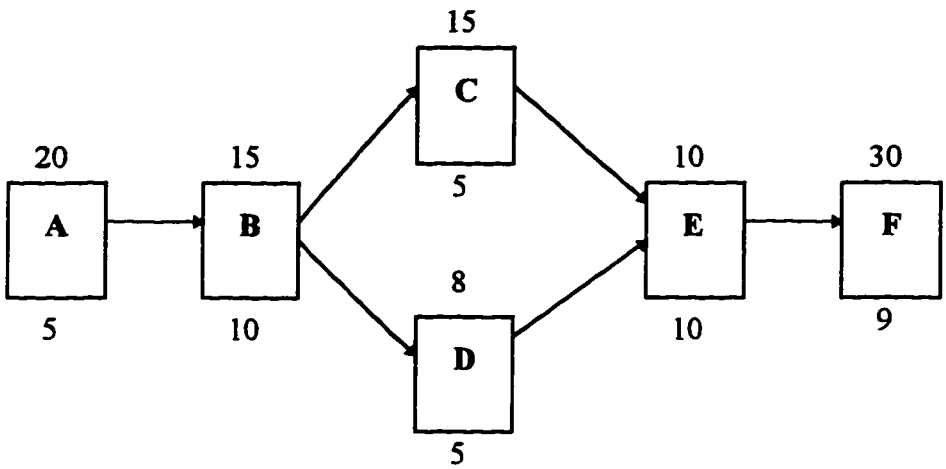


Figure 18 Tutorial map with alternative nodes

the expected arrival time for E, because there is no longer a simple linear sequence from A to E.

The prefetching policy must calculate the expected arrival time in a manner that best fits the policy goals; in this case, that means assuming the earliest possible arrival time. In the example, this would mean assuming the visitation sequence is {A, B, D, E}. However, if the student actually visits {A, B, C, E}, the policy should adjust for the later arrival time at E and any following nodes. Finally, once the student has visited either C or D, the node which is not visited is unlikely to be visited, which has implications for the management of the cache.

Removing the requirement that the visitation sequence be known in advance reinforces the importance of having clearly defined and prioritized goals for the prefetching policy, because the possibility of alternative paths means that either the system must prefetch nodes that will not be visited or that some nodes will not be prefetched prior

to the student's request for those nodes. The prioritization of the goals determines which of these options the policy should select.

The final restriction, that the visitation and retrieval times be known in advance, is clearly not feasible except in the most rigidly controlled environments. Visitation times will vary from student to student, and retrieval times will vary dependent on time of day, network and server contention, and location of the student relative to the server. The tutorial map provides a place to provide estimates of the expected visitation and retrieval times for the nodes of the tutorial. In the following section, possible methods of dealing with the variability of these times are discussed.

IMPLEMENTATION OF WWW PREFETCHING

To validate the conceptual model of prefetching presented in the previous section, prefetching was implemented in the Wtutor environment. Although Wtutor cannot know in advance the exact sequence of URLs that will be visited, it can use the tutorial map to identify the likely possible paths through the URLs of the tutorial. The tutorial map provides a partial ordering of the documents within the tutorial. As long as the student sticks to the map, Wtutor can prefetch URLs in all cases where there is sufficient time preceding the request for a URL in the map. Wtutor makes adjustments for alternative nodes and points where the student makes choices as described in the previous section.

Wtutor implements a variation of the just-in-time approach to prefetching. Doing this requires knowing the visitation times and retrieval times for the URLs in the tutorial.

Visitation times and retrieval times were added as attributes of the nodes in the map. The values stored in these attributes come from differing sources.

The retrieval time attribute for each URL is a list of historical retrieval times. As an author develops and tests a tutorial map, each time a URL is retrieved the elapsed time for retrieval of that URL is tracked and added to the list. When a student is going through the tutorial, the expected retrieval time is derived from the list of actual retrieval times. The default algorithm for estimating the retrieval time is to take the average of the actual retrieval times and add one standard deviation. This somewhat pessimistic method errs on the side of possibly overestimating the retrieval time. This is in keeping with the primary goal of the prefetch policy, which is to ensure that all nodes are prefetched before the student requests them.

One problem with this method of estimating the retrieval times is that it is based on the experience of the tutorial author, but the author's experience may not be typical of most or even any students. The retrieval times the author and the students experience can be affected by a number of factors: relative proximity to the servers providing the URLs within the tutorial, the time of day the retrievals are requested, the performance of the connection to the Internet, network contention, and caching at either the server or client. To adjust for this variability, during each session Wtutor tracks the percentage of the actual retrieval times relative to the estimated retrieval times. The differences between the actual and estimated times are tracked by server. The estimated retrieval times are then adjusted by the percentage difference between the estimated and actual times.

The visitation times must be provided as estimates by the tutorial author. Tracking the actual visitation times during development of the tutorial map would not be effective, because the author would not necessarily be reading through the documents during each visit during development of the tutorial. The author may be simply stepping through the documents to evaluate the flow of the tutorial, or switching between documents to make decisions about the order or appropriateness of the documents, or may be referring to one document while creating another. It is reasonable to expect the tutorial author to provide an estimate of the visitation time for each document in the tutorial. The author should have a target audience in mind and should have some idea of how long it will take the target audience members to go through each document.

As with retrieval times, Wtutor tracks the actual visitation times and compares the actual times with the author provided estimates. During a session, Wtutor adjusts the estimates based on the student's actual visitation times. If the student is progressing rapidly through the tutorial, Wtutor reduces the estimated visitation times so that the prefetches will be completed in time.

The timing information provided in the tutorial map may not be very accurate. The author may badly underestimate or overestimate the visitation times. The retrieval times stored in the map may not be anything like those experienced by the student. What are the possible consequences of inaccurate timing information?

At one extreme, if the estimates for visitation times are too high or the estimates for retrieval times are too low, nothing will be prefetched, because the prefetches will not

be initiated before the student requests the URLs. Under these circumstances, the student will experience the normal delays found on the World Wide Web. At the other extreme, if the estimates for visitation times are too low or the estimates for retrieval times are too high, the system will prefetch URLs much earlier than necessary. In this case some URLs may be retrieved unnecesarily if the student terminates the session without completing the tutorial. In the worst case, if the student's cache is limited, the early prefetching may cause cache overflows, which could bump unvisited documents from the cache and therefore require the system to refetch documents at request time. This would be no worse than those implementations which prefetch everything at the beginning [Softbot], [Shipman].

The implementation of prefetching in Wtutor has one significant difference from the just-in-time approach. In the Wtutor prefetching policy, the visitation times for URLs are treated as windows of opportunity for prefetching documents. What this means is that Wtutor calculates the expected arrival times for each document; calculates the LPITs for each document; then identifies for each LPIT into which document's visitation time that LPIT falls. When the identified document is requested by the student, Wtutor displays the document to the student and then immediately initiates the prefetch of the first URL whose LPIT falls within that document's visitation time. This aspect of Wtutor's prefetch policy is illustrated in Figure 19.

19a Just in time 19b Prefetch window

Figure 19 Using visitation time as a window for prefetching

In Figure 19, documents B and C need to be retrieved by the time the student completes document A. In the just-in-time model, the LPITs for C would be calculated so that the retrieval of C would complete just as the visitation time for A ends, and the LPIT for B would be calculated so that the retrieval of B would complete at the LPIT for C. In the Wtutor prefetch policy, the prefetch of B would be initiated as soon as A is displayed and the prefetch of C would be initiated as soon as the retrieval of B completes.

The negative impact of this divergence from the just-in-time policy is that some documents will be retrieved earlier than necessary. However, the benefits of this approach greatly outweigh the slight possibility that a few documents may be retrieved but not visited by the student. The primary benefit is that the impact of inaccurate estimates of visitation times are mitigated. As illustrated in Figure 20, in the just-in-time model high estimates of visitation times can lead to LPITs that are too late—documents are requested before prefetching is completed or in some cases even initiated. This leads to student experienced delays between documents. With the Wtutor approach, the prefetch of the next URL will always be initiated as soon as the current document is displayed, so even if the visitation time estimates are too high, the system is likely to have the next documents prefetched. An additional benefit of this approach is that it allows the system extra time to deal with the possible case of unavailable documents.

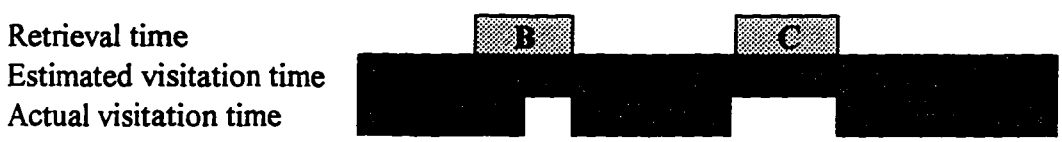


Figure 20 Impact of high visitation time estimates on just-in-time policy

PREFETCHING RESULTS

The implementation of prefetching in the Wtutor environment proved to be quite successful. With students reading through the Martin Luther King and opera tutorials with the prefetching policy in place, Wtutor was able to prefetch all but the initial documents during the visitation periods. In other words, once the initial documents of the tutorials had been retrieved, the system retrieved the rest of the documents in the tutorial map before the student requested them. In these cases, the students read through the entire tutorial, so there were no unnecessary retrievals of documents.

To more precisely measure the effectiveness of the Wtutor prefetching policy, timing studies were performed on a set of URLs. These URLs were selected as good candidates for inclusion in tutorials. To be selected, the URLs had to be cited in at least one or more "Best of" lists and be cited as an educational resource. The nineteen URLs selected were entered into tutorial maps. A group of student volunteers were instructed to step through the tutorials, using the Next menu item, and read each document retrieved. Prefetching was not activated, and the retrieval times and visitation times for each URL were measured. The results are summarized in Table 1. The contents of Table 1 are also displayed graphically in Figure 21. For each entry in Table 1, the first bar represents the visitation time for the URL while the second bar represents the retrieval time.

Not suprisingly, the retrieval time for the majority of the URLs is significantly less than the visitation time. More importantly, for most pairings of the URLs in the table, the retrieval one of the URLs could be completed during the visitation time of the other. In

Table 1 Retrieval and visitation times for selected URLs

	URL	Average Visitation Time	Average Retrieval Time	Retrieve / Visitation
1	http://hermes.astro.washington.edu/mirrors/nineplanets/mars.html	235.8	14.3	0.06
2	http://ils.unc.edu/garg/whatis1.html	19.2	13.7	0.71
3	http://ils.unc.edu/gargoyle/nathist.html	28	23.5	0.84
4	http://ils.unc.edu/gargoyle/origins.html	26.7	13.3	0.5
5	http://ils.unc.edu/gargoyle/strange.html	30.2	11.8	0.39
6	http://ils.unc.edu/gargoyle/theanswer.html	33.3	10.7	0.32
7	http://physics.nist.gov/GenInt/Time/ancient.html	108.2	20.3	0.19
8	http://physics.nist.gov/GenInt/Time/early.html	247.2	15.8	0.06
9	http://physics.nist.gov/GenInt/Time/revol.html	202.5	23.2	0.11
10	http://sln.fi.edu/franklin/inventor/inventor.html	89.8	15.5	0.17
11	http://sln.fi.edu/franklin/scientst/scientst.html	89.3	20	0.22
12	http://sln.fi.edu/franklin/statsman/statsman.html	48.3	18.2	0.38
13	http://userwww.sfsu.edu/~markd/TheFatherofLight.html	405.3	37.7	0.09
14	http://www.duke.edu/~wgrobin/ethics/	65.7	11.2	0.17
15	http://www.localnet.com/~adonis/douglass.htm	54.5	46.7	0.86
16	http://www.localnet.com/~adonis/tubman.htm	83.5	44	0.53
17	http://www.math.psu.edu/dna/twocyls/graphics.html	76.8	6.7	0.09
18	http://www.si.edu/organiza/museums/zoo/homepage/zooview/exhibits/thinktan/olp/olp.htm	49.8	48.2	0.97
19	http://www.si.edu/organiza/museums/zoo/homepage/zooview/exhibits/thinktan/olp/olpgames/foodgame/food1gam.htm	11.7	14.3	1.23
20	http://www.si.edu/organiza/museums/zoo/homepage/zooview/exhibits/thinktan/olp/olpgames/foodgame/food1key.htm	38.8	39.3	1.01

most cases, there would be a significant amount of "slack" time; that is, time where the student is still visiting the first URL of the pair and the retrieval of the second URL has been completed. This slack time is beneficial in case of retrievals that take longer than expected and in cases where more than one document must be retrieved during a single document's visitation time. The retrieval of more than one document during a document's visitation time may be necessary because of following alternative URLs or because of documents with lengthy retrieval times (as in Figure 17).

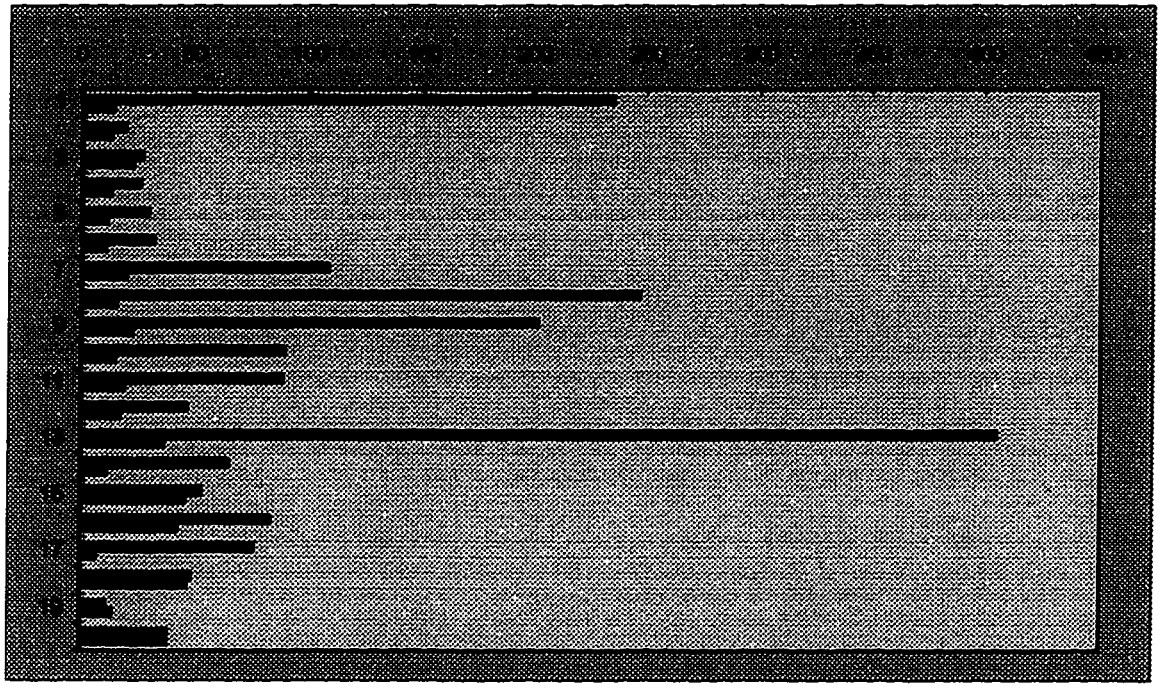


Figure 21 Visitation and retrieval times for selected URLs

There are some cases where prefetching of all documents before the student's request will not be possible. An example appears in Table 1. Entries 2-6 in the table are part of a sequence on gargoyles. The documents are visited in the order they appear in the table. The first document (entry 2 in the table) has an average visitation time of 19.2 seconds, while the second document (entry 3) has an average retrieval time of 23.5 seconds. Unless the system somehow knows in advance that the student is going to request this tutorial, it is not possible for the second document to be retrieved before the student completes the first document.

Of course, the system will initiate the prefetch of the second document as soon as the first has been retrieved and displayed, so the retrieval of the second document will be partially completed by the time the student requests it, so the system will minimize the

student experienced delay. The system will not be able to prefetch all documents if at any point in the tutorial the cumulative visitation time is less than the cumulative retrieval time for documents that must be retrieved at that point.

When measured against the goals described at the beginning of this chapter, the modified just-in-time prefetching policy compares well to other prefetching policies. The "prefetch all at initiation" policy, used in both the Browser Buddy [Softbots] and in Walden's Paths [Shipman] is effective in meeting the primary goal of having all documents fetched before the student requests them, provided that cache space is not a problem. If cache space is limited, the prefetch all at initiation policy can be counter-productive in that documents from later in the tutorial will displace earlier documents before the student reaches the earlier documents, therefore requiring a second fetch of the displaced document when the student requests it, along with a student-experienced delay while the second fetch takes place.

Even if cache space is not a problem, the prefetch all at initiation policy may fall short of the second goal—minimizing network traffic. If a tutorial has a large number of documents in it and the student is unlikely to finish the tutorial during the current session, all documents beyond the student's quitting point are fetched unnecessarily. Similarly, if there are multiple possible paths through the tutorial, where not all the documents need to be visited by the student, the modified just-in-time policy will avoid fetching many of the alternative path documents, while the prefetch all at initiation policy will fetch all the documents.

The SETS Web browser did not have very high hit rates; that is, it retrieved many documents that were not viewed by the student. The primary problem with the SETS approach was that it was attempting to predict where the student would go next. The SETS method of using the URLs within the currently displayed document as the likely set of next destinations does not prove to be very effective. However, combining the tutorial map with the SETS approach of "throttling" prefetching does have significant potential. Rather than using timing information to determine when to prefetch documents, an alternative approach would be for the prefetching policy to always stay a fixed number of steps ahead of the student. For example, the policy could be to prefetch all documents that the student could visit within the next three requests.

This alternative policy could be very effective. As long as the number of steps is not set too high, the policy would be unlikely to retrieve many more documents than the modified just-in-time policy. It would potentially break down in cases where documents with long retrieval times followed documents with short visitation times; on the other hand, it would not fail to prefetch documents because of too high visitation time estimates or too short retrieval time estimates. Another advantage of this approach is that it does not require any timing information, so it simplifies the contents of the tutorial map and the calculation of when to prefetch documents.

In the simplest prefetching policy, the system would simply prefetch the next documents in the tutorial map. This means setting the number of steps ahead of the student to one. With this simple policy, most documents would be prefetched before the

student requests them; the exceptions would be documents whose retrieval time exceeded the visitation time of the preceding document. By setting the number of steps to a higher number, the policy increases the likelihood that there will be enough visitation time to cover the retrieval times.

The timing studies also supported the approach of tracking the student's relative speed of reading. Comparing the visitation times for individual students, there are cases where students were consistently either significantly above or below the average visitation time for documents within the test tutorial.

UNAVAILABLE DOCUMENTS

There will be circumstances when URLs are unavailable. A Web server may be down for maintenance or because of problems. A Web server may be overloaded with requests and therefore unable to deliver a requested document. There may be network connectivity problems. Any of these could lead to failed requests for documents.

An additional advantage of prefetching is that it allows the system to recognize in advance that a document is unavailable, rather than at the point when the student requests the document. This gives the system time to make another try at retrieving the document, if appropriate, or it can select an alternative document and notify the student of the initial requested document's unavailability.

Wtutor provides features to deal with the availability issue. When Wtutor selects a URL to be visited next but then finds that the selected URL is unavailable, it checks whether the selected URL has an alternative URL. If there is one, Wtutor retrieves that

URL in place of the original. If an author knows that a particular URL is unreliable, the author can find or create an alternate URL If there is no alternate URL for an unavailable URL, Wtutor displays a message to the student informing them of the problem.

CHAPTER 5: SOLVING THE VALIDATION PROBLEM

Reusing existing World Wide Web materials for educational purposes presents some problems that are not encountered when an author creates a self-contained hypertext tutorial. Retrieval problems were described in the previous chapter. Another potential problem is that the owners of the reused documents may change those documents in ways not anticipated by the tutorial author. The reuse of documents requires some means of verifying that the content of the documents has not changed too significantly from the content that the tutorial selected for inclusion in the tutorial.

CHANGES TO DOCUMENTS

When a tutorial author selects documents that are owned by others for inclusion in a tutorial, the tutorial author is faced with the possibility that the document's original author may change the documents in unexpected ways. For example, the original author may decide to add significantly to a document, move it to a different location, or break it up into smaller sections. These types of changes may make the document inappropriate for inclusion in a tutorial. On the other hand, the original author may make some spelling corrections, correct a few typographical errors, or add a clarification; changes of this sort would probably not make the document inappropriate for inclusion in the tutorial.

If a tutorial system is going to encourage the tutorial author to reuse existing documents, the system needs a way to validate the new version of the document and to

determine that the document is still likely to be appropriate for inclusion in the tutorial. The following is a taxonomy of possible changes in a document:

1. Elimination of a document—this is frequently seen with documents that have been created by students, whose accounts and files are removed when the students leave school
2. Restriction of access to a document—Web servers allow the restriction of access to files or groups of files; an author may determine that access to information within a document should be limited
3. Moving of a resource to a different address—this can happen for a number of reasons: an author moves to a different Internet service provider, a different department within an organization assumes ownership of documents, etc.
4. Changes but not to the semantics relevant to the tutorial—an author may correct typographical errors, or change the wording of a document for clarity
5. Strict growing of a resource at the end of the file—frequently seen with lists, such as a list of service providers, or a list of references. The more general case is when a resource grows because items are added to lists, but not strictly at the end of the file, e.g. in alphabetical order
6. General changes to the content of the resource, independent of semantics—an author may expand the scope of a document, or break the document up into pieces

Ideally, a system that allows the reuse of materials would be able to recognize any of the above changes and deal with them appropriately. Some of the changes are explicitly defined and identified within the HTTP protocol, which specifies a set of return codes. The return codes are included in the responses from Web Servers when requests are received from client programs. The following list summarizes the relevant return codes and relates them to the list of changes above. For a complete list of HTTP return codes, see http://www.w3.org/pub/WWW/Protocols/HTTP/HTRESP.html.

- 200—URL successfully retrieved and returned. This is the code that would be returned in cases 4-6 above (see return code 304 below for the exception).
- 301—URL moved. When an author moves a resource as in case 3 above, the author can notify the Web server of the new URL for the resource. If the author has done so, the Web server returns the new URL with the 301 status.
- 302—URL found. Essentially the same as return code 301.

- 304—URL not modified. A get request for a URL can specify that the URL be returned only if the URL has been modified since a specified date. If the URL has not been modified since that date, the Web server returns the 304 status. If the URL has been modified, the current contents of the URL are returned.
- 403—Access forbidden. Document is not returned; this could be because of case 2 above.
- 404—URL not found. Could be case 1, case 2, or case 3 if the author does not notify the Web server of the new address for the URL.

The designers of the HTTP protocol recognized many of the possible changes and built features into HTTP to allow for the handling of these changes. Wtutor uses some of these features to handle the cases where a document cannot be retrieved; see the previous section on availability of documents. The HTTP protocol does not make much provision for the cases where a document can be retrieved but the contents have changed. The 304 return code can be used to determine that documents have changed, but does not distinguish between small and large changes.

A system that allows the reuse of materials could accept all documents. In fact, this is the approach used in most systems; it is assumed that if a document was appropriate when it was originally included in the tutorial, it will always be appropriate. This approach requires no validation mechanism. It also does not provide any assistance to the student when documents contained in a tutorial have changed significantly—the student has no way of knowing that the material may no longer be suitable.

In some circumstances this may be acceptable. For example, if a tutorial author only includes materials whose authors are collaborating with the tutorial author, the tutorial author can trust that any changes made to the documents are okay. Similarly, an author can choose to include only documents from trusted sources, where the tutorial

author is confident that the documents will either not be changed or only be changed in appropriate ways.

However, if the tutorial author wants the freedom to choose any materials from any source, the author is faced with the possibility that documents may be changed by the original authors and that such changes may make the documents inappropriate.

One approach to the lack of control over content of reused materials would be to not allow any change to any reused material. This philosophy would be relatively easy to implement; the tutorial map could store either the last update date or a checksum of the document text and then check this value at run-time to confirm that it has not changed (Web servers will provide the last update date for a URL upon request). Unfortunately, this approach does not allow for even minimal changes to documents. A change as small as the correction of a misspelling would lead to a document being rejected.

Ideally, a validation process would confirm that the original semantic content of the document is still present, and that the original semantic content has not been diluted excessively. The challenge is to find a validation process that can come close to achieving this ideal goal.

Given such a validation process, it could be used in many ways. Tutorial authors could use the validation process on a regular basis to identify documents that have changed significantly, and then review those documents to determine if they still meet the needs of the tutorial or if they should be replaced. The system could use the validation process when students are taking the tutorial, to determine what students should see

(either alternate documents or warnings if documents have changed excessively) and to notify the tutorial author that documents have changed excessively. The level of change at which authors would be notified could be less than the level at which students would be warned or that alternates would be used in place of original documents. This would allow authors to investigate the changes and update the tutorial before the students are impacted by the changed documents.

The validation process could also be used to monitor resources, in a manner similar to WebWatch [Surflogic]. WebWatch is a software tool that allows users to register a list of URLs and then receive notification when any of the documents represented by those URLs change. With a validation process, the user could be notified only when the documents change significantly. A database could be created to track change frequency of documents; this information could be used to determine whether documents are good candidates for inclusion in tutorials.

CONCEPTUAL MODEL OF VALIDATION

In this section, a *document* is defined as a file, where that file can be retrieved via specification of a URL. This definition does not treat included graphics as part of the document. The reference within the document to the included graphics is a part of the document.

Document *validation* is the reevaluation of a document to determine whether the document is still appropriate for inclusion in a tutorial, i.e.

$V(D) = \{accept, reject\}$

A document *signature* is a tuple of tuples, where each of the tuples in the signature is either extracted or computed from the contents of the document. A signature function maps the set of documents to a set of signatures. If D_1 and D_2 are documents, and S(D) is a signature function, then

$$D_1 = D_2 \Rightarrow S(D_1) = S(D_2)$$

however

$$\neg(S(D_1) = S(D_2) \Rightarrow D_1 = D_2)$$

Signature-based validation is a document validation approach in which the signature of a document is taken at the time the document is first included in the tutorial and then that document signature is used at run-time to determine whether the document is still appropriate for inclusion in a tutorial, i.e.

$$V(S, D) = \{accept, reject\}$$

Signature-based validation requires three steps: calculating the signature of the document, S(D), comparing the original signature to the current signature, and inferring the significance of any difference. Wtutor uses a signature-based validation approach to validate URLs.

DESIGN OF SIGNATURES

The simplest way to serve the purpose of a document signature would be to store a complete copy of the original document. This would allow comparisons to be done with existing tools such as *diff*, which could provide a detailed list of the physical changes to

the document. Given the list of physical changes, it would still be necessary to infer the significance of the changes.

This approach has the disadvantage of consuming large quantities of storage space and consuming large amounts of bandwidth either at tutorial map retrieval time or at document validation time. A tutorial could easily contain hundreds of megabytes of data, which would have to be either stored on the student's local storage medium or on a central storage medium. The space requirements could be partially mitigated by the use of compression techniques on the document text, but this would still lead to significant space and transmission requirements. What is needed is a document signature that significantly reduces storage requirements while still providing enough information about the document that useful comparisons can be made.

Another approach to creating digital signatures of documents would be for authors to agree in advance on a set of tags and values for identifying the contents of documents. This approach is used in the definition of the computer science curriculum created by the ACM [Koffman], where codes have been assigned to all the topics which should be a part of the complete computer science curriculum. This would require authors to agree in advance to provide the tags and values for the signature, which would restrict the set of documents the tutorial author could select from when creating the tutorial. It also means that the tutorial author would have to trust the original document authors to keep the tags and values for the signature up to date.

A goal of the Wtutor validation approach is to not require active participation by the authors of documents, but rather to develop a signature that will be general purpose enough to work for most World Wide Web documents.

One approach to creating a digital signature for a document would be to non-uniquely encode the content of the document. For example, a possible signature would be the number of characters in the document. This signature is small, easily calculated, and can be used to determine how much a document has grown or shrunk. It would not indicate whether the contents of the document had changed much. A similar signature is the checksum, where the characters of the document are treated as numeric values and summed. The number of digits in the checksum can be adjusted to save space or reduce the likelihood of synonyms [Horowitz].

There is a tradeoff between the space requirements of the encoding approach and the extent of document information available for comparison purposes. To provide more information about the document, the encodings could be done at some physical or logical partition of the document, such as checksums of each physical page. For purposes of comparison, it is more useful to apply the encoding to logical (document-based) partitions such as paragraphs.

Another approach to creating digital signatures would be to extract significant text from the document. This is more likely to identify semantic differences between versions of a document, where the encoding approach is more likely to identify physical

differences. The difficulty lies in determining which text is significant, because the risk of the extraction approach is that it will not recognize changes in the non-extracted material.

One method of determining the significant text would be to analyze the text with respect to a dictionary or database of documents. The dictionary or database would indicate which words or phrases are significant. Another method would be to take advantage of the structural features of the document, extracting elements such as the title or headings or links. A final method would be to require the author to specify the significant keywords or phrases within the document.

Once the composition of the digital signatures has been determined, a method of comparing signatures must be determined. This process is not completely straight forward, because different types of changes to a document may yield identical changes in the document signature. For example, an author could reword a paragraph; this would result in one checksum in the new document signature that would not match a checksum in the original signature. However, if the author deleted one paragraph and added a new one, the new signature would also have one checksum that would not match a checksum in the original signature.

The comparison process must also take into consideration whether changes in the order of elements matters. If paragraphs have been rearranged, but the content remains the same, is this a significant difference? If order matters, then the movement of an element within the document is likely to be treated as both a deletion (from the original position) and an addition (to the new position); whereas if order does not matter the movement of

an element would appear as no change at all. The comparison process should identify the common elements between two signatures, and then the changes, additions or deletions. If the elements of the signature are author specified keywords or phrases, they cannot change; the elements are either in the signature or not.

Finally, once the comparison is complete, the significance of the differences between the signatures must be determined. This could be done in a rule-based manner, e.g. "Reject a document if there are any element deletions". Alternatively a quantitative approach could be used, wherein weights are assigned to the various types of elements in the signatures and the types of differences between signatures. The method of combining these weights could be a linear, logarithmic, or polynomial equation. Once the weights have been combined into a single value, this value could be used as a distance measure, and the distance compared to threshold levels to determine whether the document has changed too much for use in the tutorial.

The need for comparing files, programs, and documents has been addressed in a number of different areas. We will not review the work done on strict file comparison, which has produced tools such as *diff*, because this work presumes the existence of two versions of a file. Rather, the focus here is on signature-based comparison methods.

DISTRIBUTED SYSTEMS

In the distributed systems arena, it is necessary to identify differences between replicated files. Research in this area has attempted to minimize the amount of data that must be passed across the network connecting the systems. This is accomplished by

transmitting digital signatures of the files; these signatures are generally checksums of the files or portions of the files.

Because of the need for generality, the only unit of structure that can be assumed for these files is the physical page [Madej]. Methods for detecting corrupted pages make a few assumptions, primarily that the files being compared are essentially the same file, with a small set of possibly corrupted pages [Barbara], [Schwarz]. The algorithms attempt to identify physically different pages for replacement purposes [Edirisooriya], [Metzner].

For purposes of validating tutorial documents, we are not overly concerned with physical differences, except that they may indirectly reflect the degree of change of the semantic content of the document.

SOFTWARE MAINTENANCE

The field of software maintenance requires program comparisons for reasons such as identifying what changes have been made in baseline software or finding clones of programs that have been changed, with the presumption that the clones will require the same changes as the original program. In [Frame], a typical line-by-line file comparison program was combined with a lexical analyzer to do comparisons of old and new versions of software. The lexical analyzer allowed the filtering out of differences in the text that would not cause differences in the code generated. In contrast, [Johnson] looks for code sections that are identical, which would indicate that one program is a clone of the other. Both these approaches require both the old and new versions of the programs.

A fingerprint based search approach is proposed in [Birch]. In this approach, a fingerprint of each reusable software component is stored, where the fingerprint is made up of features of the component such as lines of code, McCabes Cyclomatic Complexity, or the number of rewrite rules in the specification. Given a specification for a new software component, the search would compare the fingerprint of the new component with the fingerprints of existing components to identify possible matches. As noted by Birch, this approach would primarily serve to eliminate large numbers of candidates from consideration; the remaining candidates would still need to be compared with the new specification at a detailed level.

INFORMATION RETRIEVAL

The field of information retrieval deals with finding relevant documents in large databases of documents. The user specifies a query, such as a set of keywords or a phrase that is desired. The system attempts to find documents in the database that match the query. To speed the process of searching, information retrieval systems build a large index of the documents in database. Documents are represented by a vector which contains the significant terms found in the document, with weights assigned to those terms. The weight assigned to each term reflects both the frequency of the term within the document and the uniqueness of the term to the document as opposed to other documents in the database.

When a query is specified, the query is evaluated in the same manner as the documents; e.g. the terms of the query are identified and weighted. The resulting vector is then compared to the vectors in the database index. In [Salton], the measure of how well

the query and a document match is the inner product of the query vector and the document vector. In [Robertson], the measure of how well the query and a document match is the sum of the weighted terms that match. In either case, the documents are ranked according to how well they match the query and some subset of the highest ranked documents are presented to the user.

This approach comes fairly close to accomplishing the desired validation. However, it does have a few flaws for our purposes. It requires the analysis of a database of documents to establish what terms are significant and to determine the weights that should be assigned to these significant terms. Additionally, the storage requirements for the signatures can be quite large [Callan]; a document's signature could be as large or larger than the original document. Finally, the match measures derived from these processes are designed for comparative purposes (e.g., A is a better match than B); they do not have an absolute value for purposes of determining that A is a good enough match.

WTUTOR SIGNATURES

Wtutor takes advantage of the structural information provided in HTML documents to extract meaningful document signatures. In Wtutor, document signatures are composed of three elements: the checksums of the paragraphs of the document, the headings of the document, and any keywords the tutorial author chooses to specify. Figure 22 shows a sample signature. Before extracting the signature, Wtutor normalizes the document by converting all the text to lower case and reducing each contiguous group of whitespace characters to single space characters.

URL	http://www.cs.washington.edu/research/metip/tutor.form.image.html
Paragraphs	(19316 47247 21696 47247 21696 33122 37527 38582)
Headings	(quantization sampling rate applying linear filters tutorial image request form)
Keywords	(NIL)

Figure 22 Sample Wtutor signature

Other possibilities were considered for inclusion in the signature, such as the title of the document or the links contained within the document. The paragraph checksums, headings, and keywords were chosen as the elements that best provide information about the document.

PARAGRAPH CHECKSUMS

Wtutor assumes that most URLs represent HTML documents and takes advantage of the HTML structural information to break the document up into paragraphs [1]. For each of the paragraphs, a checksum is computed; the checksum is the sum of the numeric value of each character within the paragraph, modulo 100000. For non-HTML documents, such as GIF or JPEG images, the entire document is treated as a single paragraph, so a single checksum is computed and stored for the document.

[1] The <p> tag is used in HTML to indicate the start of a paragraph. Wtutor uses the <p> tag to identify paragraphs. This does not necessarily break a document into all the logical paragraphs within the document, because HTML authors do not always use the <p> tag in every instance where one is logically implied; for example, authors do not always insert a <p> tag before or after headings, because the heading generally implies the paragraph break.

Keeping the paragraph checksum allows Wtutor to make inferences about the degree of physical change a URL has experienced. Given the paragraph count and checksums, Wtutor can determine what percentage of paragraphs have changed and whether additions or deletions have been made. There is a very slight possibility of not finding actual changes because of checksum synonyms, but the number of digits in the checksums is large enough that the likelihood is extremely small.

LIST OF HEADINGS IN DOCUMENT

Paragraph checksums provide information about the degree of physical change within a document, but do not provide much information regarding the semantic content. Wtutor assumes that the headings within a document are significant indicators of the semantic content of the document. This assumption might not be true for all HTML documents; however, the set of documents that must be addressed is restricted to those documents that a tutorial author might select for inclusion in a tutorial. This means that it can be assumed that the documents are well written, in which case the headings should in fact contain relevant semantic information.

In HTML documents, authors identify headings with tags that indicate not only that the enclosed text is a heading, but also indicate the heading level. For example, <h1>High Level Heading</h1> indicates a heading at the highest level. Wtutor extracts all headings from the document for inclusion in the document signature. Some massaging of the headings is done to reduce extraneous HTML markup information that frequently occurs in headings.

KEYWORD BASED SIGNATURES

The first two elements of Wtutor's digital signatures can be calculated automatically without any intervention by the author. However, an author may feel that these two elements do not provide enough information regarding one or more documents within a tutorial. For this reason, Wtutor also allows the tutorial author to specify one or more key words, on either a full tutorial or individual document basis. When an author specifies such keywords, Wtutor searches the documents to see if any of the keywords appear; any keywords that do appear in the document are included in that document's signature.

WTUTOR SIGNATURE COMPARISON METHOD

Wtutor stores document signatures in the tutorial map. The tutorial author can run a procedure at any time to update the signature of any URLs in the tutorial map. When a student is using the tutorial, Wtutor validates each URL by comparing the digital signature stored in the map with the document's current digital signature. The comparison yields a numeric distance value, which indicates how much the document has changed.

For each element of the signature, Wtutor infers whether there have been additions, changes, or deletions. When comparing elements of two document signatures, Wtutor uses the following method: First any exact matches are removed. If the exact matches do not account for all entries, then Wtutor determines how many changed entries there are, by presuming that each remaining entry in the new signature that has a corresponding entry in the original signature represents a changed entry. Finally, if the new

signature has fewer entries than the original, then the difference is counted as the number of deletions; if the original signature has fewer entries than the new, the difference is counted as the number of additions.

Once the number of changes, additions, and deletions in each category have been determined, they are combined according to the function in Figure 23. To allow for documents of different lengths, the function normalizes by the number of items of each type within the original document. This normalization recognizes that the scale of changes matters—adding five paragraphs to a document that originally had two paragraphs is probably much more significant than adding five paragraphs to a document that originally had fifty paragraphs.

dist = checksum dist + heading dist + keyword dist

checksum dist = checksum weighting factor *
(#chgs * chg_weight + #adds * add_weight + #deletes * del_weight) / #checksums

heading dist = heading weighting factor *
(#chgs * chg _weight + #adds * add_weight + #deletes * del_weight) / #headings

keyword dist = keyword weighting factor *
#deletes * del_weight / #keywords

Figure 23 Document signature distance function

The distance function also recognizes that some types of differences can be more significant than others by including a weighting factor for each type of difference. For example, it is probably more significant if a heading has been deleted from a document (implying that some segment of information is no longer provided by the document) than if

a heading has been added (additional information is generally not a bad thing, unless so much is added that the original content is overwhelmed by the new).

The distance function also allows for the different elements of the signature to be weighted differently. A tutorial author may feel that the headings of a document are more significant than the paragraph checksums, since the headings provide more of an indication of the semantic content of the document. Another tutorial author may feel that no changes of any sort should be allowed, and choose to weight all elements equally. The Wtutor distance function provides the author the flexibility of choosing these weights.

DEFAULT WEIGHTS

Wtutor assigns a default set of values to the weighting factors in the distance function. The default values are shown in Figure 24. These default values reflect several assumptions regarding tutorial materials and document signatures. The greater weight assigned to headings and keywords relative to paragraph checksums reflect the assumption that the headings and keywords provide more information about the semantic content of the document than the checksums. Similarly, the greater weight assigned to deletions relative to changes reflects the assumption that it is more significant if material has been removed from a document than if the material has been changed; a similar assumption applies to additions relative to changes and deletions.

(defvar *dist-del-weight* 4.0)	weighting of deletions
(defvar *dist-chg-weight* 2.0)	weighting of changes
(defvar *dist-add-weight* 0.5)	weighting of additions
(defvar *dist-par-weight* 0.2)	weighting of paragraph checksums
(defvar *dist-head-weight* 0.4)	weighting of headings part of thumbprint
(defvar *dist-key-weight* 0.4)	weighting of keywords part of thumbprint
(defvar *URL-warning-distance* 0.25)	distance at which warning is displayed
(defvar *URL-reject-distance* 0.50)	distance at which URL is rejected

Figure 24 Default distance function weighting factors

The default weights have a significant impact on the distance function. Because of the values chosen, the distance function is not symmetric, i.e. the distance from A to B is not necessarily equal to the distance from B to A. Consider the documents in Figure 25 and Figure 26 as an example. If we take sample document B as the original document and compare it to A, we see that a paragraph has been added. The original information is still there, the document is not significantly larger than before, so it is likely that A would be considered not too different from B. Conversely, if we take A as the original document and compare it to B, we find that a significant piece of information (the second paragraph) is no longer in the document. In this case, we would consider B to be significantly different from A and therefore would expect the distance measure from A to B to be greater than the distance measure from B to A.

<p>Start your outdoor workout *indoors* with several minutes of stretching. This will reduce risk of injury and make you less prone to overdress.
<p>Wear a hat to moderate your body temperature. Forty percent of body heat is lost through the head. Taking your hat on and off will keep you comfortable.

Figure 25 Sample document A

<p>Start your outdoor workout *indoors* with several minutes of stretching. This will reduce risk of injury and make you less prone to overdress.

Figure 26 Sample document B

Once the distance for a document has been calculated, Wtutor compares that distance to two defined levels: a warning level and a reject level. If the distance from the original signature to the current signature is less than the warning level, the document is simply displayed to the student. If the distance is greater than the warning level but less than the reject level, the document is displayed and a warning message is provided to the student indicating that the document has changed since its inclusion in the tutorial. If the distance is greater than the reject level, Wtutor first checks whether the document has an alternative. If an alternative has been specified in the tutorial map, that document is retrieved and displayed instead of the rejected document. If there is no alternative document, the rejected document is displayed to the student along with a message that warns the student that the document has changed significantly and may not be appropriate for the tutorial.

ANALYSIS OF WTUTOR SIGNATURE COMPARISON METHOD

Figure 27 restates the distance formula for Wtutor signature comparisons. In this statement of the formula, D represents the distance. In the first line of the formula, W_{ck} represents the weighting factor for checksums, $n_{ck,chg}$ represents the number of changed

$$D = W_{ck} * (n_{ck,chg} * W_{chg} + n_{ck,add} * W_{add} + n_{ck,del} * W_{del}) / n_{ck} \quad \text{(checksums)}$$
$$+ W_{hd} * (n_{hd,chg} * W_{chg} + n_{hd,add} * W_{add} + n_{hd,del} * W_{del}) / n_{hd} \quad \text{(headings)}$$
$$+ W_{ky} * (n_{ky,del} * W_{del}) / n_{ky} \quad \text{(keywords)}$$

Figure 27 Document signature distance function

checksums, W_{ch} represents the weighting factor for changes, $n_{ck,add}$ represents the number

of added checksums, W_{add} represents the weighting factor for additions, $n_{ck,del}$ represents the number of deleted checksums, and n_{ck} represents the number of checksums. The factors for headings and keywords are similarly named.

Consider an original document (M) and a later version of the document (N), where paragraphs have been added to M to create N; that is, $n_{ck}(N) > n_{ck}(M)$.

Then $n_{ck,add} = (n_{ck}(N) - n_{ck}(M)) > 0$ and $n_{ck,chg} \leq n_{ck}(M)$ and $n_{ck,del} = 0$

So $W_{ck} * (n_{ck,chg} * W_{chg} + n_{ck,add} * W_{add} + n_{ck,del} * W_{del}) / n_{ck}$

$= W_{ck} * (n_{ck,chg} * W_{chg} + n_{ck,add} * W_{add}) / n_{ck}$

This value is unbounded because the $n_{ck,add}$ factor is unbounded. This reflects the fact that there is no limit to the material an author can add to a document. Now consider the case where no paragraphs have been added to the document, that is $n_{ck}(N) \leq n_{ck}(M)$.

Then $n_{ck,add} = 0$ and $n_{ck,chg} \leq n_{ck}(M)$ and $n_{ck,del} = (n_{ck}(M) - n_{ck}(N))$
and $n_{ck,chg} + n_{ck,del} \leq (n_{ck}(M)$

So $W_{ck} * (n_{ck,chg} * W_{chg} + n_{ck,add} * W_{add} + n_{ck,del} * W_{del}) / n_{ck}$

$= W_{ck} * (n_{ck,chg} * W_{chg} + n_{ck,del} * W_{del}) / n_{ck}$

$\leq W_{ck} * W_m * (n_{ck,chg} + n_{ck,del}) / n_{ck}$ where $W_m = \max (W_{chg}, W_{del})$

$\leq W_{ck} * W_m$

This value is bounded. If an author does not add new paragraphs, but only modifies or deletes paragraphs within the earlier version of the document, the maximum distance is bounded by the weighting factors. The analysis for the headings component of the distance

function is essentially the same as for the checksum component. The analysis for the keywords component is simpler for two reasons. Changes to keywords are not possible (if a keyword is changed it is no longer a keyword); and keyword additions are ignored. For the deletions, $n_{ck,del} \leq (n_{ck}(M)$.

So $W_{ky} * (n_{ky,del} * W_{del}) / n_{ky}$

$\leq W_{ky} * W_{del}$

Combining the limits for the cases where no additions are made yields

$D \leq (W_{ck} + W_{hd} + W_{ky}) * W_m$ where $W_m = \max (W_{chg}, W_{del})$

TESTING SIGNATURE COMPARISONS RESULTS

Several experiments were conducted to test the effectiveness and usefulness of the Wtutor document validation method. These experiments used documents from the World Wide Web along with author initiated changes and sets of controlled changes to those documents to test how appropriately Wtutor inferred the significance of the changes. On average, the document signatures were only about 5% as large as the original documents. This satistied one of the goals of the signature-based validation approach, in that the signatures required significantly less storage than the documents.

CONTROLLED CHANGES TO SELECTED DOCUMENTS

In the first experiment, a controlled set of common changes was applied to selected documents. The HTML documents in the selected set were chosen as good examples of documents that an author might choose to include in a tutorial. The URLs for

these documents are listed in Figure 28. These documents are well-written documents that contain useful information. All of these URLs were found in existing lists of useful resources. They are good samples of candidates for inclusion in a tutorial.

1	http://www.cc.gatech.edu/gvu/nsf-ws/report/Authoring.html
2	http://www.w3.org/hypertext/WWW/Provider/Etiquette.html
3	http://wk122.nas.nasa.gov/NAS/Education/TeacherWork/Ozone/Modeling_lesson.html
4	http://ipl.sils.umich.edu/ref/RR/COM/computer.html
5	http://www.ncsa.uiuc.edu/demoweb/html-primer.html
6	http://sands.psy.cmu.edu/ACT/tutor/tutor-info.html

Figure 28 Selected WWW documents

The following list describes each document in the set of selected WWW documents:

1. Authoring tools—from *Research Priorities for the World-Wide Web*, the Report of the NSF Workshop Sponsored by the Information, Robotics, and Intelligent Systems Division held October 31, 1994 in Arlington, VA.
2. Etiquette for information providers—from the World Wide Web Consortium.
3. Modeling_lesson—from a resource file on Stratospheric Ozone Depletion.
4. Computers Ready Reference—from the Internet Public Library.
5. HTML primer—from the NCSA.
6. The Advanced Computer Tutoring Project at CMU.

The documents were also chosen to provide a range of document characteristics. Table 2 provides statistical details about each of these documents. For the selected documents, a set of common changes was developed and applied. A list of the changes is provided in Table 3. Because of the difference in content of each document, it is not possible to make identical changes to each document. However, the changes were designed to be as similar as possible.

Table 2 Selected WWW document characteristics

URL name	Bytes	Lines	Paragraphs	Headings
Authoring.html	5109	42	10	4
Etiquette.html	4087	120	3	7
Modeling_lesson.html	13693	226	26	4
computer.html	10151	223	17	7
html-primer.html	41066	1197	115	45
tutoring.html	3704	89	7	4

The set of changes listed in Table 3 are subjectively ranked in increasing order by degree of change to the original document. Admittedly, the degree of change that each of these changes represent is difficult to rank, especially when the change is described abstractly rather than concretely and specifically. However, the ranking in Table 3 is an effort to judge the relative significance of the listed changes.

Testing Wtutor's evaluation of these changes required several steps. First, the signature of each original document was calculated and stored in a tutorial map. Then for each document, each of the controlled changes was applied to a copy of the original document, resulting in a new version of the document. For each new version, Wtutor calculated the new signature and calculated the distance from the original version to the modified version. The results of these calculations are shown in Figure 29.

The distance measures for the versions of the documents are proportionate to the size of the documents. With a fixed number of changes, we would expect that the larger the document, the smaller the distance would be between the original and the modified version. The results bear this out. The distance measures for the html-primer document are consistently the lowest values; the html-primer document is by far the largest of the sample

Table 3 Controlled changes applied to selected WWW documents

Ver	Description of change
0	original version of document
1	three spelling corrections, one case change
2	three spelling corrections, one case change, four added words
3	one added paragraph
4	one added heading
5	one added paragraph, one added heading
6	four added paragraphs
7	three spelling corrections, one case change, four added words, one added heading
8	three spelling corrections, one case change, four added words, one added paragraph, one added heading
9	four added paragraphs, one added heading
10	three spelling corrections, one case change, four added words, four added paragraphs
11	four added paragraphs, two added headings
12	three spelling corrections, one case change, four added words, four added paragraphs, one added heading
13	three spelling corrections, one case change, four added words, four added paragraphs, two added headings
14	three spelling corrections, one case change, four added words, one added paragraph
15	one deleted paragraph
16	three spelling corrections, one case change, four added words, one deleted paragraph
17	one added paragraph, one deleted paragraph
18	three spelling corrections, one case change, four added words, two deleted paragraphs
19	two deleted paragraphs
20	one added paragraph, two deleted paragraphs
21	one deleted heading
22	three spelling corrections, one case change, four added words, one deleted heading
23	two deleted paragraphs, one deleted heading
24	three spelling corrections, one case change, four added words, two deleted paragraphs, one deleted heading
25	four added paragraphs, two added headings, one deleted paragraph
26	four added paragraphs, two added headings, two deleted paragraphs
27	one added paragraph, one deleted heading
28	one added paragraph, two deleted paragraphs, one deleted heading
29	four added paragraphs, two added headings, one deleted heading
30	four added paragraphs, two added headings, two deleted paragraphs, one deleted heading

documents. The distance measures for the etiquette document are generally the highest; the etiquette document is one of the smallest documents, and has the fewest paragraphs and headings of the documents in the sample set. The measures for the authoring and tutoring documents are comparable, as are the measures for the modeling-lesson and the

computer documents; these pairs of documents are also similar in size and number of paragraphs and headings (refer to Table 2 Selected WWW document characteristics).

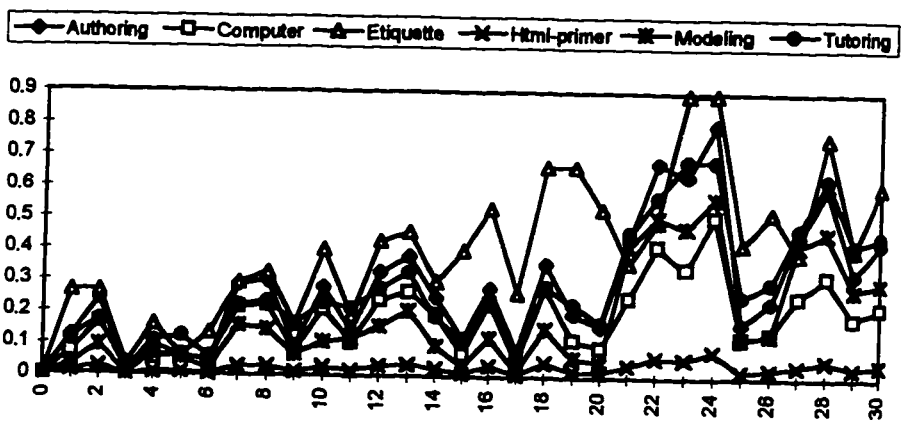


Figure 29 Distances from original documents to modified versions

The distance measures for the versions of the documents match up well with the default rejection and warning levels. For most of the documents, only a few of the versions exceed the reject level. The versions that are generally rejected (versions 22, 23, 24, 28) all include deleted headings as well as many other changes to the original document. There are more versions that exceed the default warning level. These versions are primarily those that include deletions of headings or paragraphs along with a large number of changes relative to the overall size of the document. The largest document, the html-primer, has no versions that exceed even the warning level. At the other extreme, one third of the etiquette document versions exceed the reject level and one half the versions exceed the warning level. This reflects both the small size and the unusual structure of the document (it only has three paragraphs, so deleting even one paragraph is a highly significant change to the document).

The overall slope of the lines for each document tend to rise, indicating that overall the distance measure does reflect the subjective judgment of degree of change. However, there is significant variation in the relative rankings of the versions. There are two primary reasons for this variation. The first reason is that this method of comparing documents cannot distinguish between large and small changes within paragraphs. Using Version 2 from Table 3 as an example, the case change is ignored, but because each of the three spelling corrections and four added words were done in separate paragraphs, the net effect was seven changed paragraphs. Even though these small changes within the paragraphs are unlikely to significantly affect the semantics of the paragraphs, they count the same as complete rewrites of the paragraphs in question. Because only the paragraph checksums are recorded, Wtutor can only recognize that a paragraph changed, not how much the paragraph changed.

The second reason for the variation is that the comparison method collapses the addition and deletion of items of the same type as a change. For example, if a paragraph is added to a document and another paragraph is deleted, the comparison method would find one checksum that differs between the original signature and the new signature. If the order of headings and paragraphs were counted as significant, this collapsing of adds and deletes would be less likely.

Figure 30 illustrates the impact on the distance measure of these factors. In the chart in Figure 30, the different versions of the computer document are shown ranked in increasing order by the distance measure. In this ordering, the versions that included the

spelling changes and added words (versions 1, 2, 7, 8, 12-14, 16, 22, 24) all end up much higher in the calculated rankings than in the subjective ranking. Similarly, the versions that included both the addition and the deletion of paragraphs or headings (versions 17, 20, 25, 26, 29, 30) all ended up much lower in the calculated rankings.

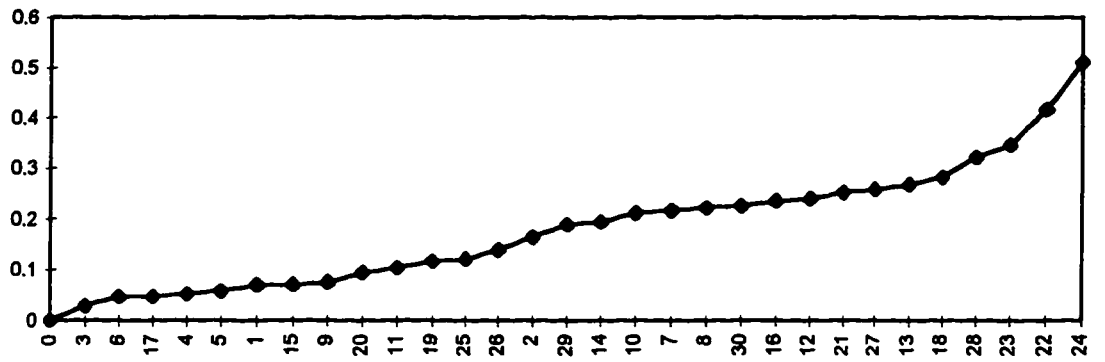


Figure 30 Versions of computer document ordered by distance measure

DOCUMENT LIFE CYCLE

The testing described above used a set of artificial changes introduced into the sample documents specifically for testing purposes. A better test of the Wtutor approach to validating documents is to see how well it measures historical changes that actually were made to documents. There were three cases in which documents were tracked over a period of time and the distance measures calculated for the varying versions of those documents.

The first case is a Web page describing Wtutor. There were five different versions of this document. The initial version (version 0) is a brief (72-line) description of Wtutor. Version 1 is essentially the same document, with a bit of rewording done. Figure 31 is a listing of the *diff* output comparing version 1 to version 0. Version 2 adds introductory paragraphs to version 1; the *diff* output comparing version 2 to version 1 is shown in Figure 32. Version 3 is a User's Guide to Wtutor, which includes most of the material in version 2. This document was significantly larger than the earlier versions. The *diff* output is not included here, as it is 498 lines long. Finally, version 4 of the document goes back to version 2, but adds links to sections of the User's Guide to Wtutor. The *diff* output comparing version 2 to version 4 is shown in Figure 33.

The distance measures for each of the versions of the Wtutor document are shown in Table 1. These distance measures are the result of comparing the signatures of the

```
29a30
> The tutorial map defines the tutorial to Wtutor.
44c45
< may be appropriate, but the author must still evaluate the candidates to
---
> may be appropriate. The author must still evaluate the candidates to
56c57
< feature is provided, so that the author can  specify groups of URLs
---
> feature is provided, so that the author can specify groups of URLs
64,65c65,66
< that is, if URL A and URL B are alternatives, the student does not need
< to view B when the student has already viewed A.
---
> that is, if URL A and URL B are alternatives, the student would not need
> to view B if the student had already viewed A.
```

Figure 31 Output of diff comparison of version 0 and version 1

```
8a9
>
9a11,42
> The World Wide Web is a rich source of material that can be used for
> educational purposes. Advantages of reusing these existing materials
> are avoiding the
> duplicated effort of creating new versions of material that is already
> in existence, using up-to-date information, and letting the student see
> the original source materials. In addition this approach permits tutorial
> authors to focus on adding value by organizing, commenting on, and adding
> to existing information.
> <p>
> The World Wide Web provides a vast store of globally available information
> from a variety of sources. Government agencies such as the National
> Institute of Health, NASA, and the Smithsonian operate multiple Web
> servers that allow access to data, reports, art and analysis. Most
> universities and colleges have Internet access and these institutions,
> their departments, faculty, staff and students have set up
> Web servers to provide information about the schools and their research.
> Organizations such as CERN make specifications and proposals available via
> the Web. Businesses and individuals operate Web sites for self-promotional
> purposes or for love of a subject. All of these different resources can be
> used to find materials for inclusion in tutorials.
> <p>
> Using existing materials from the WWW for educational purposes presents
> a number of problems that must be dealt with. The first problem is the
> filtering problem: with all the material out there on the Web, how does
> the student know which materials are relevant to the chosen subject area?
> And of those relevant materials, which best present the subject at a
> level appropriate for the student? Once the set of appropriate materials
> has been selected the student is confronted with the navigation problem:
> given the set of appropriate materials, where should the student start?
> In what order should the student view those materials? When is the student
> finished?
> <p>
```

Figure 32 Output of diff comparison of version 1 and version 2

versions to the signature of version 0. The distance measures do not exceed the warning level for versions 1, 2 and 4. This is a desirable result, as the changes in these versions are

```
128a129,135
> <ul>
> <li>
> <a href="http://www.cs.washington.edu/homes/dbj/sources/dbj/using_wt.html">
> Using Wtutor</a>
> <li>
> <a href="http://www.cs.washington.edu/homes/dbj/sources/dbj/using_mt.html">
> Using MTutor</a>
```

Figure 33 Output of diff comparison of version 2 and version 4

Table 4 Distance measures for versions of Wtutor document

Version	Distance	Description
0	0.000	Original version; brief description of Wtutor
1	0.145	Slight rewording of version 0
2	0.209	Addition of introductory material to version 1
3	2.718	User's Guide to Wtutor
4	0.245	Back to version 2, with addition of links to User's Guide

minor and do not significantly change the semantic content of the document. Version 3 exceeds the reject level. Again, this is a desirable result. While the semantic content of the earlier versions is included in version 3, the document contains so much more material that it is not really comparable to the other versions.

SAMPLE TUTORIAL

The second case in which Web documents were tracked over time was the sample tutorial about Dr. Martin Luther King. Many of the documents that make up this tutorial were described in chapter 1. Although the tutorial was created primarily as an illustration of Wtutor's capabilities, it also served as a good test of the validation approach, because several of the documents that were selected for inclusion in the tutorial changed after the tutorial was originally created.

About a month after the creation of the tutorial, a running of the validation report for the tutorial showed the chronology document as having a distance measure of 0.007. This small distance is well below the warning level, so it did not have an impact on the tutorial. Upon investigation, the reason for the distance was found to be a correction of one of the dates in the chronology.

Two weeks later, another running of the validation report showed very large distance measures for several documents. All the documents from one site had distances in the 1.9-2.3 range, well above the reject level. Further investigation revealed that the owner of the documents had moved to a different site. The original site provided a note that the documents had moved, but did not provide direct links to the new locations. This required updating the tutorial map to point to the new locations for the documents.

In both of these cases, the validation method worked as desired. In the first case, the change to the document was recognized as small enough that the student was not even informed of the difference. In the second case, Wtutor recognized that the documents returned by the original server were not at all what the tutorial author wanted and notified the student of that fact. Without the validation, the student would not necessarily known that the documents returned were not what the tutorial author intended.

Two weeks later, a running of the validation report showed one document with a distance measure of 0.62, again above the reject level. Upon inspection, the document was found to have been updated by the owner. In the course of correcting several typos in the document, the author had significantly reformatted the document. In this case, the

validation method returned a "false positive". The text portion of the document had not changed significantly, but the differences in markup were interpreted by the signature comparison as significant document changes. A student going through the tutorial would have received a warning message that the document might no longer be appropriate for the tutorial, but the document would still have been displayed to the student.

AUTHOR VOLUNTEERED EXAMPLES

In the final case of tracked Web documents, Web document authors were solicited via Usenet postings for participation in the experiment. Prospective participants were asked if they had Web documents that they were planning to update. Dr. Hubert Partl of the Universitaet fuer Bodenkultur in Vienna, Austria volunteered to participate and provided a list of the URLs for his documents. Dr. Partl's documents make up an HTML handbook.

A tutorial map that contained each of the URLs was created and document signatures were taken. Copies of each document were saved at the same time as the document signatures were taken so that actual changes to the documents could be tracked. During a six month study, periodic validation reports were taken to test whether the documents had changed since the initial document signatures were taken. At the same time as the validation reports were done the current versions of the documents were retrieved and compared to the original copies, so that the actual changes could be reviewed and the appropriateness of the resulting distance measure could be evaluated. The URLs for the volunteered set of documents are listed in Table 5, along with two

distance measures for each document. The intermediate distances are from compares done about two months into the measurement period; the final distances are from the compares done at the end of the measurement period.

Table 5 Volunteered WWW document distance measures

URL	Intermediate Distance	Final Distance
http://www.boku.ac.at/htmleinf/hein.html	0.22	0.39
http://www.boku.ac.at/htmleinf/hein1.html	0.30	0.44
http://www.boku.ac.at/htmleinf/hein2.html	0.28	0.50
http://www.boku.ac.at/htmleinf/hein3.html	0.21	0.29
http://www.boku.ac.at/htmleinf/hein4.html	0.10	0.43
http://www.boku.ac.at/htmleinf/hein5.html	0.27	0.44
http://www.boku.ac.at/htmleinf/hein6.html	0.18	0.21

How well do these distance measures reflect the changes that were made to the documents during the measurement period? The first observation is that only one version of the documents exceeds the default rejection level. This is a desirable result, as most of the changes to the documents fell into the categories of markup changes, spelling changes or corrections, rewordings for clarity, or minor additions. For most of the documents, the final versions exceed the warning level. This is primarily due to the number of minor changes that were made to these documents. The final version of hein2.html just reaches the reject level. The reason that this document exceeds the reject level is that it includes some deletions of headings and paragraphs.

The distance measures for the intermediate and final versions of these documents seem appropriate for the differences between these updated versions and the original versions of the documents. Overall, the Wtutor approach to validating documents provides a useful method of verifying that documents still fit within the tutorial.

CHAPTER 6: EVALUATION

The Wtutor environment enables the reuse of World Wide Web materials in tutorials. Using the Wtutor environment, tutorial authors can create tutorials from their own Web documents as well as existing documents found on the Web. Once a tutorial map has been created, the Wtutor environment can be used to lead students through the tutorial. The experience of going through the tutorial is enhanced by the prefetching of documents, which reduces or even eliminates wait time for the student. Finally, the Wtutor environment ensures that the retrieved materials are still appropriate for the tutorial via the validation function.

The prefetching and validation features are needed because the Wtutor approach to creating Web-based tutorials uses the current version of all documents. The advantages and disadvantages of using the current version of documents were discussed in chapters 4 and 5; fortunately, in enough cases the advantages outweigh the disadvantages. This is fortunate because there is not necessarily a good alternative. Unless the owner has given permission, copying a document and presenting it with a different URL for the tutorial could be considered a form of plagiarism, copyright infringement, or both. Presenting a copy of a document as the actual document (that is, with the original URL) when the document with that URL may have changed could be a misrepresentation of the document. This means that if a tutorial is going to reuse existing documents from the

Web, the current version of those documents should be retrieved as the student is going through the tutorial.

Given the set of features that the Wtutor environment provides, how well does the Wtutor environment measure up to an "ideal" Web-based tutorial? Although the Wtutor environment provides the basic functionality needed for a tutorial system, along with some additional beneficial features (such as prefetching and validation), there are several areas in which improvements could be made. These areas include Wtutor's model of student interaction, platform and browser dependence, prefetching and caching algorithms, and tutorial authoring interface.

WTUTOR MODEL OF STUDENT INTERACTION

There is a wide range of possible methods of using the Web for educational purposes. One method would be purely exploratory. In this method, students would be simply turned loose with a subject area and a browser, possibly with a starting point or a search tool, and it would be up to the students to find the relevant materials on the Web. At the other extreme, a totally structured method might have an explicit set of documents, with all movement between documents restricted by the browser and no visiting of documents outside the defined domain. The guided tour method used in the Wtutor system takes the best of these two extremes, providing a defined domain and visitation sequence but also allowing students to take exploratory sidetrips.

The original intent of Wtutor was to encourage exploratory learning by students with nonintrusive guidance from Wtutor. In this exploratory model, the tutorial map

provides the layout of the domain of interest and Wtutor acts as a guide to that domain. Students would proceed through the domain of the tutorial primarily by following links within the documents. Wtutor would highlight links to indicate good choices for the next document to visit. The student would only request help from Wtutor when the student could not decide where to go next, as in the situation where the student comes to a document that has no links to any unvisited materials within the domain. This model did not work as well as hoped, for a number of reasons.

A tutorial is expected to provide a fairly high degree of structure. Generally, a tutorial is expected to be linear, repeatable and consistent. The exploratory model used by Wtutor does not fit these expectations, because students follow their own path through the materials. How the student proceeds can differ too. Using Wtutor, from one document the student may proceed by following an imbedded link within a document, from another the student may follow an explicit "Next" link at the end of the document, from another the student may click on a button, and from another the student may use the "Next" option under the Tutor menu. One effect of this is that it is not necessarily clear to the student which visited documents are part of the tutorial. Both Footprints [Nicol] and Walden's Paths [Shipman] make it very clear if a document is part of the current path or not. These implementations also use explicit mechanisms for all movement along the path.

The consistent model of student interaction provided by the Footsteps and Walden's Paths path implementations is more appropriate for tutorials. The consistency helps remove the uncertainty regarding whether documents are part of the tutorial or not

and reduces the cognitive load on the student, because the student does not have to analyze the links to decide which to select. The Wtutor interface could be changed to better fit the path model without requiring changes to the tutorial map method of describing the tutorial. An example of the type of changes needed is that Wtutor marks a document as visited when the student visits the document, regardless of whether the visit came within the flow of the tutorial. For tutorials, this is probably inappropriate, because documents should be seen in the specified order, even if they have been seen before. Also, the current student interaction model does not allow the tutorial author to specify that a document should be visited more than once. An author may desire to repeat a visit for dramatic effect, or to clarify a point, or to reinforce an important concept.

Another reason that the exploratory approach does not work well in practice is the lack of links between materials created by different authors. Authors tend to create islands of information. While authors often provide good linkage within their own body of work, the links to others' work tend to be more haphazard. Links from an author's materials to related materials authored by others are often not embedded within the content of the documents, but are in separate lists of related materials.

A final reason that the guided, exploratory approach does not work well has to do with the evolution of authoring on the Web. An initial idea behind Wtutor providing guidance was that links to documents that were ready to be visited would be highlighted through use of special colors. In this manner of guidance, the students would gain information from the content and the context of the links that would help illuminate the

relationship between the documents. In the early days of the Web, authors used the standard markup features provided by HTML. As the popularity of the Web exploded, more browsers became available and developers of the browsers began adding features to differentiate their browsers from others. Authors now have much more control over the appearance of Web pages.

Unfortunately, the concept of highlighting breaks down when document authors can and do control the display of links. For example, the Wtutor default color for highlighting links was red; today, some documents on the Web use red as the default color for all link displays, while other documents specify red as the color for links that have already been visited. In the first case, the student would be led to believe that all the links on the current document are good choices to visit next. In the second case, the student would be led to believe that the documents that are good choices to visit next are those that have already been visited. In general, colored links cannot be depended on as guides for navigation purposes.

One way in which tutorial maps significantly differ from other path or guided tour mechanisms is that alternative paths are allowed. This feature again goes back to the idea of exploratory learning—students would follow the links that interested them and in the course of doing so would end up taking one of the alternative paths. In the more directed structure of the tutorial, alternative paths become less likely to be used, and raise some questions regarding how the system should deal with them. When the student comes to a

point where alternative paths can be taken, which path should be chosen? In the exploratory model, the student chooses. In the tutorial model, the system must choose.

Wtutor treats alternative paths as equally valid choices. However, the tutorial author may prefer one path over another. The tutorial author may provide an alternative path only for those circumstances when documents in the preferred path are unavailable. The tutorial map should provide a means of identifying explicitly whether alternative paths are equally good choices or if one is preferred over the other.

If the paths within the tutorial are made more linear, with explicit indications of which paths are preferred, the likelihood of successful prefetching becomes higher. For one thing, the more linear the paths are the easier it is to identify which documents to prefetch. For another, linearity reduces the number of documents that might be visited next, so at any point within the tutorial the number of documents requiring prefetching is reduced. This is especially true when prefetching is being done several steps ahead of the student's current position within the tutorial, as in the throttled prefetching approach.

PLATFORM AND BROWSER DEPENDENCE

Ideally, a student would be able to initiate a Web-based tutorial from the student's chosen browser on whatever platform the student happens to be using. Rather than needing a specific browser on a particular platform, the student should be able to select a URL that would initiate the tutorial within the student's browser. This is the primary advantage of the Footprints and Walden's Paths implementations. Because Footprints and Walden's Paths use scripts on an intermediate server to implement paths, the paths can be

followed using most Web browsers. But this has two implications. First, all links within the documents must be modified to invoke scripts on the path server. This means that the link displays do not match those of the original document. Second, retrieval of documents is done in two phases: from the source server to the path server and then from the path server to the browser.

The Wtutor and BRIO [Roscheisen] implementations display documents without modification (except in the case of BRIO annotations, which are explicitly identified as modifications to the underlying document). In the Wtutor environment, the documents are retrieved directly by the browser, so the two-phase retrieval is eliminated. Unfortunately, this comes at the cost of using a specific browser in both of these implementations.

A primary area for future research is whether it is possibile to create a tutorial system that can be launched from a browser in such a way that the student can continue to use that browser to read through the documents of the tutorial, and the tutorial system can track the student's progress and provide interactive guidance to the student.

PREFETCHING AND CACHING ALGORITHMS

The tutorial map stores pointers (URLs) to the documents that make up the tutorial. It also stores document signatures for validation purposes. An alternative approach would be to actually store copies of the documents in the map. This caching of documents would essentially move the prefetching function up from runtime to creation time. Wtutor does do some document caching, but only at runtime. Once a document has

been retrieved in Tmosaic it is kept until the cache space is needed for storage of newly retrieved documents.

As mentioned at the beginning of this chapter, storing copies of documents could lead to concerns of copyright, plagiarism, or misrepresentation. However, if the owner of a document explicitly gives permission to make copies of the document, there are benefits to storing a copy of the document.

If the tutorial author is not concerned with possible updates to the documents, then storing copies of the documents would eliminate the need for any validation function, because the documents could be stored as the tutorial author found them at tutorial creation time. It would also eliminate the problem of unavailable documents—anytime the tutorial map could be retrieved, then the documents making up the tutorial would be retrieved. Even more promising would be a hybrid strategy. In this strategy, a copy of the original document would be stored. At runtime, Wtutor would attempt to prefetch the original document. If the prefetch were successful, the retrieved document would be displayed at the appropriate time to the student. If prefetching were unsucessful, Wtutor could display the copy of the document. This strategy would provide ideal alternative documents, in that the backup document would be the version of the document that the tutorial author originally selected for inclusion in the tutorial. The backup copy could also be used if the retrieved version of the document were found to have changed significantly from the original. Of course, this strategy would significantly increase the storage requirements for tutorial maps.

When prefetching, Wtutor treats all documents alike. Documents are prefetched when Wtutor determines that they are soon likely to be requested by the student. With the increasing popularity of audio and video documents, it may be necessary to reevaluate this approach. Audio and video documents are usually extremely large and take a long time to retrieve. On one hand, this means that prefetching these documents has a high payback to the student in that long delays can be avoided. On the other hand, these documents could consume all of the local cache or consume a lot of disk space. If the documents are indeed visited by the student, then prefetching is a big win, but if the student does not visit the documents, a high price has been paid in terms of network and cache resource consumption.

Some documents may not benefit from being prefetched. For example, RealAudio audio segments are played back as they are retrieved. Prefetching a RealAudio document could lead to the document playing back while the student is still visiting other documents. This would be counter productive in terms of distracting and confusing the student. Even if the playback could be prevented, the streaming nature of such documents means that prefetching probably does not provide a significant benefit.

TUTORIAL AUTHORING INTERFACE

The focus in the development of Wtutor was on the student interface and functionality of the system. An unfortunate consequence of this is that the Wtutor tutorial authoring interface is not very user-friendly. Wtutor uses a command line interface where the commands are Lisp function names. While these commands are fairly mnemonic, they

are also lengthy and require exact syntax. In addition, the map display options are text listings of the map contents. Identifying the relationships between entries in the map is difficult to do. Users have commented on how much they like the functionality of Wtutor, but they have found it difficult to create tutorial maps and have needed assistance from the programmer to create tutorials. The most common request from users has been for a graphical display of the tutorial map.

A graphical user interface (GUI) for the tutorial authoring part of Wtutor would be much more user-friendly and usable. Graphical depiction and editing of the tutorial map would make it much easier to use and understand, both by authors and students. A graphical depiction of the tutorial map which indicated current position within the map would help students understand the scope and structure of tutorial, as well as provide a measure of the student's progress through the tutorial.

FUTURE WORK

Future work on Wtutor will primarily revolve around addressing the shortcomings described in the preceding sections of this chapter, that is, developing a more consistent student interaction model, creating a platform and browser independent version of Wtutor, refinement of the prefetching and caching algorithms, and the creation of a graphical user interface to Wtutor, especially for tutorial authoring. In addition, further work will be done to enhance Wtutor's basic tutorial support, explore retrieval and prefetching issues, and improve on the current validation technique.

ENHANCEMENTS TO BASIC TUTORIAL SUPPORT

In Wtutor tutorial maps the basic unit of granularity is the document as represented by a URL. However, many documents have anchors within them that identify the beginning of sections of the documents. A tutorial author may want to include only a subsection of a document in a tutorial. This is not currently possible in Wtutor.

Wtutor currently assumes that once a URL has been visited that it has been read and understood. This is a very simple model of student comprehension. One possible improvement would be to include expected visitation intervals for each document within the tutorial and then measure the students' visitation intervals at the documents. If a student spent either significantly longer or shorter time visiting a document, the system could intervene. What type of intervention would be appropriate?

RETRIEVAL AND PREFETCHING ISSUES

The timing studies conducted for this research were from a fairly small sample. Results from larger samples would be useful to determine whether the results are typical of what students would encounter on the Web in terms of the relationships between retrieval times and visitation times. Studies on a greater number of URLs from a broad selection of Web servers would be very useful.

One assumption of the Wtutor implementation of prefetching is that the retrieval times encountered during creation of the tutorial map are representative of the retrieval times that students will encounter while taking the tutorial. This may not be true, if the students are geographically distributed in locations far removed from that of the tutorial

author. Testing the validity of this assumption requires doing retrieval timings from a number of sites distributed over broad geographic regions.

On the other hand, it may be reasonable to eliminate the need for storing and using timing information altogether. We will be testing the "throttled" prefetching policy, where rather than using timing information to determine when URLs should be retrieved, the system will simply stay a predetermined number of steps ahead of the student. How many steps ahead are required to ensure prefetching of all or most of the documents in a tutorial? Can this number be kept small enough that the system does not prefetch an excessive number of unused documents?

Finally, the implementation of prefetching in Wtutor is a simple single threaded implementation that does not allow interruption by the student. Multithreaded prefetching would be more efficient and effective, and should allow the student full interaction with the current document display in parallel with the retrieval of other documents.

ENHANCEMENTS TO VALIDATION

There are several possible ways in which the signature comparison method used by Wtutor could be improved. The current comparison method looks for exact matching between the normalized versions of the headings within a document. While this method finds all the unchanged headings, it does not recognize the degree of change of those headings which do change. A second step could be added to the heading comparison process to find the degree of match between the changed headings, so that small changes would not be treated as being as significant as large changes.

The current signature comparison method also treats all headings as being of equal significance. However, HTML allows for multiple levels of headings. A change in an <h1> heading is probably more significant than a change in an <h6> heading. A measure that took into account the heading level would probably be a better measure of the significance of identified changes.

The signature extraction method and signature comparison method in Wtutor are designed to be generally appropriate. However, a tutorial author may reason to use different signatures and comparison methods for particular documents. One way of doing this would be to allow the specification of particular signature methods for documents. A special case where this would be appropriate is in different types of documents. The default signature extraction and comparison methods assume HTML documents; there are many other types of documents that could be included in a tutorial. The signatures and comparison methods for GIF or JPEG documents could borrow from those used in search systems [Jacobs]. Similar document-type specific signatures and comparison methods could be developed for non-HTML text documents such as PDF or PostScript documents.

The possibility of smaller granularity of documents was mentioned earlier. If this were to be implemented, then a corresponding change in the signature extraction method would be appropriate, such that only the relevant subsection of the document would be extracted from to create the signature.

One of the common changes that occurs to documents is that they are moved. When this happens, the documents receive a new URL. Often, the original document is

replaced with a document containing a link to the new location. It could be possible for Wtutor, upon finding a document that had changed extremely, to search the retrieved version of the document for any links and then to retrieve the linked to documents and compare them to the original document signature. If the comparison were very close, Wtutor could assume that the document had moved and that the new location had been found. This could conceivably lead to self-correcting maps, with Wtutor updating the map to point to the new location of the document.

Assuming this could be done, a further step would be to identify documents that have been split up. Upon finding a document that had significantly reduced in size, Wtutor could search for links in the document, follow those links, and then test the retrieved documents for subsets of the original documents signature elements. This would require a sophisticated method of partial matching between the original document and the new documents that potentially comprise the content of the original.

BIBLIOGRAPHY

[Allinson89] Allinson, L. and Hammond, N. A Learning Support Environment: The Hitch-Hikers Guide. In *Hypertext: Theory Into Practice*, ed. McAleese, R. 1989, Norwood, NJ: Ablex Publishing Corporation. pp. 62-74.

[Allinson92] Allinson, L. Learning Styles and Computer-Based Learning Environments. In *Proceedings of the Fourth International Conference on Computers and Learning.* Wolfville, Nova Scotia, Canada, pp. 61-73, 1992.

[Barbara] Barbara, D. and Lipton, R.J. A randomized technique for remote file comparison. In *Proceedings of the Ninth International Conference on Distributed Computing Systems*. Newport Beach, CA, USA, pp. 12-19, 1989.

[Barber] Barber, Philip. Authoring for Multi-Media CAL. In Barber, Philip (ed.), *Multi-Media Computer Assisted Learning*, New York: Nichols Publishing, pp. 210-224.

[Bender] Bender, A., Edman, A. and Sundling, L. A combined knowledge and hypermedia system to attain educational objectives. In *Proceedings of the sixth IFIP World Conference on Computers in Education*, Chapman & Hall, London, 1995, pp. 67-74.

[Benest] Benest, I. D. An Alternative Approach to Hypertext, *Educational and Training Technology International.* Vol. 28, No. 4, Nov. 1991. pp. 341-346.

[Bennet] Bennet, E. (ed). *The Society of Text: Hypertext, Hypermedia, and the Social Construction of Information.* Cambridge, Mass: The MIT Press, 1989.

[Bessiere] Bessiere, C., and Vacherand-Revel, J. Graphical Authoring of Multimedia Courseware. In McDougall, A., and Dowling, C. (eds.), *Computers in Education.* New York: Elsevier Science Publishers B. V., pp. 871-876.

[Bestavros]Azer Bestavros, "Speculative Data Dissemination and Service to Reduce Server Load, Network Traffic and Service Time for Distributed Information Systems" *Proceedings of ICDE'96: The 1996 International Conference on Data Engineering*, New Orleans, Louisiana. March 1996. Also available at http://cs-www.bu.edu:80/faculty/best/res/papers/icde96.ps.

[Bestavros]Azer Bestavros, "Using speculation to reduce server load and service time on the WWW" in *Proceedings of CIKM'95: The Fourth ACM International Conference on*

Information and Knowledge Management, Baltimore, Maryland. November 1995. Also available at http://cs-www.bu.edu:80/faculty/best/res/papers/cikm95.ps.

[Bestavros]Azer Bestavros, "Demand-based document dissemination to reduce traffic and balance load in distributed information systems" in *Proceedings of the 1995 Seventh IEEE Symposium on Parallel and Distributed Processing*, San Antonio, Texas. October 1995. Also available at http://cs-www.bu.edu:80/faculty/best/res/papers/spdp95.ps.

[Birch] Birch, C.M. and Nevill, D.G. Software metric fingerprints. In *Proceedings of the 11th BCS IRSG Research Colloquium on Information Retrieval*, Huddersfield Polytech, Huddersfield, UK, pp. 36-49. 1989.

[Bloom] Bloom, B. S. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, 13 (1984) pp. 3-16.

[Botafogo] Botafogo, R. A. and Shneiderman, B. Identifying Aggregates in Hypertext Structures. In *Proceedings of the Hypertext '91 Conference*, ACM, New York, pp. 63-74, 1991.

[Brusilovsky] Brusilovsky, P. L. A Framework for Intelligent Knowledge Sequencing and Task Sequencing. In *Proceedings of the Second International Conference on Intelligent Tutoring Systems*. Montreal, Canada, pp. 499-506, 1992.

[Buckley] Buckley, C., Allan, J. and Salton, G. Automatic Routing and Retrieval Using Smart: TREC-2. *Information Processing and Management*. Vol 31, No. 3, pp. 315-326, 1995.

[Bush] Bush, V. As We May Think. *Atlantic Monthly*. Vol 176, No. 1, pp. 641-649, 1945.

[Callan] Callan, J., Croft, W.B. and Broglio, J. TREC and TIPSTER Experiments with INQUERY. *Information Processing and Management*. Vol 31, No. 3, pp. 327-343, 1995.

[Carrier] Carrier, C. A., and Jonassen, D. H. Adapting Courseware to Accommodate Individual Differences. In Jonassen, D. H. (ed.), *Instructional Designs for Microcomputer Courseware*. Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 203-226.

[Chan] Chan, T.-W. Curriculum Tree: A Knowledge-Based Architecture for Intelligent Tutoring Systems. In *Proceedings of the Second International Conference on Intelligent Tutoring Systems*. Montreal, Canada, pp. 140-147, 1992.

[Delaney] Delaney, P. and Landow, G. P. *Hypermedia and Literary Studies*. Cambridge, Mass: The MIT Press, 1991.

[Dowling] Dowling, D. Authoring for English Literature in Hypercard: Where in the World is Blake's Tyger? *Hypermedia*, Vol. 4, No. 3, 1992. pp. 171-196.

[Edirisooriya] Edirisooriya, S. and Edirisooriya, G. A signature efficient solution for remote file comparison. In *Proceedings of the Sixteenth Annual International Computer Software and Applications Conference*. Chicago, IL, USA, pp. 149-154, 1992.

[El Hani] El Hani, O. and Gouarderes, G. Standardized Architecture for Integrated Open Courseware. In *Proceedings of the Fourth International Conference on Computers and Learning*. Wolfville, Nova Scotia, Canada, pp. 198-211, 1992.

[Frame] Frame, M. A lexical comparison program to simplify the maintenance of portable software. In *Proceedings of the Conference on Software and Maintenance*. Scottsdale, AZ, USA, pp. 348-350, 1988.

[Frasson] Frasson, C., Gauthier, G., and McCalla, G. I. (eds.), *Intelligent Tutoring Systems: Proceedings of the Second International Conference on Intelligent Tutoring Systems, ITS '92*. Berlin, Germany: Springer-Verlag, 1992.

[Gloor] Gloor, P. A. CYBERMAP Yet Another Way of Navigating in Hyperspace. In *Proceedings of the Hypertext '91 Conference*, ACM, New York, pp. 107-121.

[Guinan] Guinan, C. and Smeaton, A.F. Information Retrieval from Hypertext Using Dynamically Planned Guided Tours. In *ECHT '92: Proceedings of the ACM Conference on Hypertext*, ACM, New York, pp. 122-130, 1992.

[Halasz 1988] Halasz, F. G. Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems. *Communications of the ACM*, 31 (7): 836-852, July 1988.

[Halasz 1990] Halasz, F. G. and Schwartz, M. The Dexter Hypertext Reference Model. In *Proceedings of the Hypertext Workshop*, National Institute of Standards and Technology, Gaithersburg, Md. NIST Special Publication 500-178, March 1990, pp. 95-133. Reprinted in *Communications of the ACM*, Vol. 37, No. 2, pp. 30-39.

[Hall] Hall, W., Hutchings, G. and White, S. Breaking down the barriers: an architecture for developing and delivering resource based learning materials. In *Proceedings of the sixth IFIP World Conference on Computers in Education*, Chapman & Hall, London, 1995, pp. 623-633.

[Hara] Hara, Y., Keller, A. M. and Wiederhold, G. Implementing Hypertext Database Relationships through Aggregations and Exceptions. In *Proceedings of the Hypertext '91 Conference*, ACM, New York, pp. 75-90.

[Hendrikx] Hendrikx, K., Duval, E. and Olivie, H. Hypermedia for open and flexible learning. In *Proceedings of the sixth World Conference on Computers in Education*, Chapman & Hall, London, 1995, pp. 349-361.

[Hicks] Hicks, D. L. *A Version Control Architecture for Advanced Hypermedia Environments*. Doctoral Dissertation, Texas A & M University, Dec. 1993.

[Horowitz] Horowitz, E. and Sahni, S. *Fundamentals of Data Structures*. Potomac, Maryland: Computer Science Press, Inc. 1976.

[Instone] Instone, K., Teasley, B. M., and Leventhal, L. M. Empirically-based re-design of a hypertext encyclopedia. In *Proceedings of ACM INTERCHI '93 Conference on Human Factors in Computing Systems*, ACM, New York, pp. 500-506.

[Jacobs] Jacobs, C.E., Finkelstein, A., and Salesin, D.H. Fast Multiresolution Image Querying. In *Computer Graphics* Proceedings, Annual Conference Series, 1995, ACM SIGGRAPH, pp. 277-286.

[Johnson] Johnson, J.H. Substring Matching for Clone Detection and Change Tracking. In *Proceedings of the IEEE International Conference on Software and Maintenance*. Eds. Müller, H.A. and Georges, M. Sept. 19-23, pp. 348-350, 1994.

[Jonassen] Jonassen, D. and Mandl, H. (eds.), *Designing Hypermedia for Learning*. Berlin, Germany: Springer-Verlag, 1989.

[Jordan] Jordan, D. S., et al. Facilitating the Development of Representations in Hypertext with IDE. In *Proceedings of the Hypertext '89 Conference*, ACM, New York, pp. 93-104.

[Koffman] Koffman, E. B., Stemple, D. and Wardle, C. E. Recommended curriculum for CS2, 1984 : a report of the ACM curriculum task force for CS2. *Communications of the ACM* 28, 8 (Aug. 1985), pages 815-818.

[La Passardiere] La Passardiere, B. de and Dufresne, A. Adaptive Navigational Tools for Educational Hypermedia. In *Proceedings of the Fourth International Conference on Computers and Learning*. Wolfville, Nova Scotia, Canada, pp. 555-567, 1992.

[Ladd] Ladd, B., Capps, M., Stotts, D., and Furuta, R. "Multi-head/Multi-tail Mosaic: Adding Parallel Automata Semantics to the Web," World Wide Web Journal, O'Reilly and Associates Inc., vol. 1 (Proc. of the 4th International WWW Conference, Boston, December 11-14, 1995), pp. 433-440. This paper appears on-line at http://www.w3.org/pub/Conferences/WWW4/Papers/118/.

[Laurel] Laurel, Brenda, Oren, Tim, and Don, Abbe. Issues in Multimedia Interface Design: Media Integration and Interface Agents. In *CHI '90 Conference Proceedings*, (April 1990), pp. 133-139.

[Landow] Landow, G. P. Popular Fallacies About Hypertext. In Jonassen, D. H. and Mandl, H. (eds.), *Designing Hypermedia for Learning*. Berlin: Springer-Verlag, pp. 39-59, 1990.

[Mackay] Mackay, W. E. Tutoring, Information Databases, and Iterative Design. In Jonassen, D. H. (ed.), *Instructional Designs for Microcomputer Courseware*. Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 327-346.

[Madej] Madej, T. An application of group testing to the file comparison problem. In *Proceedings of the Ninth International Conference on Distributed Computing Systems*. Newport Beach, CA, USA, pp. 237-243, 1989.

[Marchionini] Marchionini, G. and Crane, G. Evaluating Hypermedia and Learning: Methods and Results from the Perseus Project. *ACM Transactions on Information Systems*. Vol 12, No. 1, 1994, pp. 5-34.

[Marshall] Marshall, C. C. and Irish, P. M. Guided Tours and On-Line Presentations: How Authors Make Existing Hypertext Intelligible for Readers. In *Proceedings of the Hypertext '89 Conference*, ACM, New York, pp. 15-26.

[Maurer] Maurer, H. Why Hypermedia Systems Are Important. In *Proceedings of the Fourth International Conference on Computers and Learning*. Wolfville, Nova Scotia, Canada, pp. 1-17, 1992.

[Metzner] Metzner, J.J. Efficient replicated remote file comparison. *IEEE Transactions on Computers*. Vol 40, No. 5, 1995, pp. 651-660.

[Nicol] Nicol, D., Smeaton, C. and Slater, A. F. Footsteps: Trail-blazing the Web. In *Proceedings of the 1995 World Wide Web Conference*, 1995.

[Nicolson] Nicolson, R. I., and Tomlinson, P. USHIR: A Knowledge-based Hypermedia System. *Hypermedia*, Vol. 3, No. 1, Winter 1991, pp. 1-33.

[Nkambou] Nkambou, R. and Gauthier, G. Use of WWW Resources by an Intelligent Tutoring System. In *Proceedings of ED-MEDIA 96—World Conference on Educational Multimedia and Hypermedia*. Boston, Mass., USA, pp. 527-532, 1996.

[Nott] Nott, M. W., Riddle, M. D. and Pearce, J. M. Enhancing traditional university science teaching using the World Wide Web. In *Proceedings of the sixth IFIP World Conference on Computers in Education*, Chapman & Hall, London, 1995, pp. 623-633.

[Olimpo] Olimpo, G., Chioccariello, A., Midoro, V., Persico, D., Sarti, L., Tavella, M., Trentin, G. On the Concept of Databases of Multimedia Learning Material. In McDougall, A., and Dowling, C. (eds.), *Computers in Education*. New York: Elsevier Science Publishers B. V., pp. 431-436.

[Padmanabhan] Padmanabhan, V.N. "Improving World Wide Web Latency", *Technical Report UCB/CSD-95-875*, University of California, Berkeley, CA, USA, May 1995. Also available at http://http.cs.berkeley.edu/~padmanab/papers/masters-tr.ps.

[Papert] Papert, S. *The Children's Machine: Rethinking School in the Age of the Computer*. New York: HarperCollins Publishers, Inc, 1993.

[Pinckney] Tom Pinckney, M. Frans Kaashoek, and Joshua A. Tauber, "Dynamic Documents: Extensibility and Adaptability in the WWW", *Electronic Proceedings of the Second World Wide Web Conference '94: Mosaic and the Web*, available at http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/DDay/pinckney/dd.html.

[Recker] Recker, M. M. and Perolli, P. Student Strategies for Learning Programming from a Computational Environment. In *Proceedings of the Second International Conference on Intelligent Tutoring Systems*. Montreal, Canada, pp. 382-394, 1992.

[Robertson] Robertson, S. E., Walker, S. and Hancock-Beaulieu, M.M. Large Test Collection Experiments in an Operational, Interactive System: Okapi at TREC. *Information Processing and Management*. Vol 31, No. 3, pp. 345-360, 1995.

[Roscheisen] Roscheisen, M., Mogensen, C. and Winograd, T. Beyond Browsing: Shared Comments, SOAPs, Trails, and On-line Communities. In *Proceedings of the 1995 World Wide Web Conference*, 1995.

[Schwarz] Schwarz, T., Bowdidge, R.W. and Burkhard, W.A. Low cost comparisons of file copies. In *Proceedings of the Tenth International Conference on Distributed Computing Systems*. Paris, France, pp. 196-202, 1990.

[Shipman] Shipman, F. M., Marshall, C.C., Furuta, R., Brenner, D.A., Hsieh, H., and Kumar, V. Creating Educational Guided Paths over the World-Wide Web. In *Proceedings of ED-TELECOM 96—World Conference on Educational Telecommunications*. Boston, Mass., USA, pp. 326-331, 1996.

[Softbots] Softbots, Inc. http://www.softbots.com/bb_home.htm.

[Steere] Steere, D.C. Using Dynamic Sets to Speed Search in World Wide Information Systems. *Technical Report CMU-CS-95-174*, Carnegie Mellon University, Pittsburgh, PA, USA, March 1995. Also available at ftp://reports.adm.cs.cmu.edu/usr/anon/1995/cmu-cs-95-174.ps.

[Surflogic] Surflogic LLC. WebWatch Products. See http://www.surflogic.com/ww.1.x/products . html.

[Thierry] Thierry, B. and Andre, P. An Object Oriented Approach To Produce Educational Hypermedia Software. In *Proceedings of the Fourth International Conference on Computers and Learning*. Wolfville, Nova Scotia, Canada, pp. 111-123, 1992.

[Thuring] Thuring, M., Haake, J. M. and Hannemann, J. What's Eliza doing in the Chinese Room? Incoherent hyperdocuments--and how to avoid them. In *Proceedings of the Hypertext '91 Conference*, ACM, New York, pp. 161-177.

[Touch]J.D. Touch, "Defining `High Speed' Protocols : Five Challenges & an Example That Survives the Challenges," IEEE Gigabit Networking Workshop, Toronto, 1994. Also in IEEE JSAC., special issue on Applications Enabling Gigabit Networks, Vol. 13, No. 5, June 1995, pp. 828-835 (also ISI/RS-95-408). Also available at ftp://ftp.isi.edu/pub/hpcc-papers/touch/jsac.gb_enab_apps.ps.Z and at http://www.isi.edu/~touch/pubs-latency.html.

[Trigg86] Trigg, R.H. and Weiser, M. TEXTNET: A Network Based Approach to Text Handling, *ACM Transactions on Office Information Systems*, Vol. 4, No. 1 (January 1986), 1-23.

[Trigg88] Trigg, R.H. Guided Tours and Tabletops: Tools for Communicating in a Hypertext Environment, *ACM Transactions on Office Information Systems*, Vol. 6, No. 4, (October 1988), 398-414.

[Trigg91] Trigg, R.H. From Trailblazing to Guided Tours. In Nyce, J.M. and Kahn, P. (eds.), *From Memex to Hypertext: Vannevar Bush and the Mind's Machine*. Academic Press, Inc., San Diego, pp. 353-367, 1991.

[Van Marke] Van Marke, K. Instructional Expertise. In *Proceedings of the Second International Conference on Intelligent Tutoring Systems*. Montreal, Canada, pp. 234-243, 1992.

[Vassileva] Vassileva, J. Dynamic CAL-Courseware Generation Within an ITS-Shell Architecture. In *Proceedings of the Fourth International Conference on Computers and Learning*. Wolfville, Nova Scotia, Canada, pp. 581-591, 1992.

[Venezsky] Venezsky, Richard, and Osin, Luis. *The Intelligent Design of Computer-Assisted Instruction*. New York: Longman Publishing Group, 1991.

[Zeiliger] Zeiliger, R., Reggers, T. and Peeters, R. Concept-Map Based Navigation in Educational Hypermedia: a Case Study. In *Proceedings of ED-MEDIA 96—World Conference on Educational Multimedia and Hypermedia*. Boston, Mass., USA, pp. 714-719, 1996.

[Zellweger88] Zellweger, P. Active paths through multimedia documents. In *Document Manipulation and Typography*, J.C. van Vliet (ed.), Proceedings of the International Conference on Electronic Publishing, Document Manipulation and Typography, Nice (France), April 20-22, 1988. Cambridge University Press, 1988, pp. 19-34.

[Zellweger89] Zellweger, P. Scripted Documents: A Hypermedia Path Mechanism. In *Proceedings of the Hypertext '89 Conference*, ACM, New York, pp. 1-14.

VITA

David B. Johnson was born in Seattle, Washington on April 6, 1957. After alternating between Seattle Pacific University and Oregon State University for several years, he graduated from Oregon State University in 1981 with a B.S. in Computer Science and a B.S. in Business Administration. Upon graduation, he accepted a position with Pacific Northwest Bell, which later became U S WEST. In 1988 he completed the Master of Software Engineering program at Seattle University. Since he did not yet have too many degrees, he enrolled at the University of Washington, where he received his M.S. degree in Computer Science and Engineering in 1992 and his Ph.D. in Computer Science and Engineering in 1997.