Domain Name Based Visualization of Web Histories in a Zoomable User Interface

Rajiv Gandhi   Girish Kumar
Ben Bederson   Ben Shneiderman
Department of Computer Science
University of Maryland, College Park
Maryland, USA
{gandhi, girish, bederson, ben}@cs.umd.edu

Abstract

Users of hypertext systems like the World Wide Web (WWW) often find themselves following hypertext links deeper and deeper, only to find themselves "lost" and unable to find their way back to the previously visited pages.

We have implemented a web browser companion called Domain Tree Browser (DTB) that builds a tree structured visual navigation history while browsing the web. The Domain Tree Browser organizes the URLs visited based on the domain name of each URL and shows thumbnails of each page in a zoomable window.

A usability test was conducted with four subjects.

1   Introduction

The use of the WWW has increased dramatically in the last few years. The availability of browsers for multiple computing platforms, many of them available at no cost allows even novice computer users with limited resources to make use of the wide range of services and information available on the internet.

However, navigating the WWW is difficult for users. After following a number of links, users often have trouble revisiting a page that was previously viewed. According to the 6th GVU survey, 13.4% of subjects report not being able to find pages recently visited [8].

The same survey also found that while 42% of the pages were visited using the Back-Button, only a meager 0.1% of the page accesses used the history list. This shows that the pages were revisited with a high frequency, however the history list is hardly used. This suggests that the history mechanisms in the current browsers are not appealing to users. Some of the shortcomings of the common history mechanisms are as follows. First, whenever a user follows a branch point, a large part of the history is lost. Second, the history list is textual and page titles may lack cues needed to find a particular page. Third, the history list is cumbersome to use. A user must pull down a menu before finding and following the desired entry.

The difficulty in revisiting previously viewed pages may discourage users from engaging in exploratory behavior. We believe that the addition of a graphical history view would help users navigate the WWW more easily.

We have built a visualization tool, the Domain Tree Browser, which keeps track of all visited pages within a domain in the form of a tree. It creates a node in the tree for every visited page and puts a thumbnail image of the web page on it. Our system also provides basic sorting and searching capability on domains. We believe that this tool will help the users in revisiting already visited pages and will give them a sense of context.

2   Related Work

Several projects have investigated web usage visualization before. WebMap is a browser extension that shows a graphical relationship between web pages [5]. Each page is represented by a small circle that can be selected to display the actual page. Links between pages are colored to indicate information such as whether it is a link to a different server or whether the destination page has already been read. Pad-Prints [6] is a tool which visualizes the pages visited by a user in the form of a single tree. It takes screen grabs of visited pages and puts them on the nodes of the tree. MosaicG is a modified version of Mosaic version 2.5 that provides a two-dimensional view of the documents visited by a user in a session [1]. The Graphic History View presents titles, uniform resource locators (URLs), and thumbnail images of the documents a user has visited in a session. The graphical layout is a two-dimensional tree built from left to right with

visual cues. As graphs get large, the user has the options of: zooming out for a smaller representation of all documents in the tree, and condensing branches of the tree that are no longer of interest. Footprints is a prototype system created to help people browse complex web sites by visualizing the paths taken by users who have been to the site before [11]. These paths are shown as a graph of linked document nodes, with the links color-coded to visualize the frequency of use of the different paths. The map does not represent all the possible paths within a site or all the possible links a user could follow from any given page. Rather, the map shows what people actually did in the represented site over the sample time.

Our work differs from the ones described here in several important ways. First, we do not attempt to construct a map of the web site. We construct a tree of the thumbnails of the pages visited in a domain. Second, unlike PadPrints [6], we do not have a single tree modeling the entire history. Instead, we organize the visited pages into different domains and maintain one tree for each domain.

3 Domain Tree Browser

3.1 Features

The Domain Tree Browser(DTB) is a personal web history visualization tool. It is intended to be used as a browser companion. It receives events from the web browser whenever hyperlinks on a web page are clicked and uses those events to create and maintain personal web histories. It constructs a hierarchy as the user traverses the links, which is in contrast with pre-building the hierarchy for a web-site, as WebMap [5] and several other systems do.

DTB automatically maintains web histories, with minimal effort from the user. The tool organizes the visited URLs based on web-site domains. It's zoomable user interface automatically resizes thumbnails to fit the window.

Figure 1 shows a screenshot from DTB. DTB is divided into two parts - the panel on the left displays the names of all the domains visited so far. This panel is referred to as the *domain panel*. The *tree panel* is to the right of the domain panel and displays the tree visualization of the visited URLs of the domain selected on the left panel. Each tree represents the visits made by a user in one domain. Each node in the tree corresponds to a visited URL. A node is a rectangle which contains the screen grab of the web-page it represents. The tree hierarchy is displayed in a top-down manner. The rightmost frame is the browser window where the web pages are displayed.

A tree corresponding to a domain maintains the user's last visited node in that domain and marks it in green. For ease of description, let the tree displayed in the tree panel be called *current tree*, the domain corresponding to the tree be called *current domain* and the last visited node as *current node* in that domain. When a hyperlink is clicked on the web page, there are two cases. In the first case, the user has already visited that page and hence a node corresponding to that page already exists in some tree. The node is made the current node in the corresponding tree and receives a green border. If this tree is not already the current tree then the tree is made current and is displayed on the tree panel and the corresponding domain becomes the current domain. In the second case, the user has not been to this page before. In this case, a new node is created. If the user has already visited the domain, this new node is added as the child of the last node visited (current node) in that domain. If not, a new tree corresponding to the domain is created, and the new node is added as the only visited child of this tree. The tree panel is tightly coupled with the browser window. By this, we mean that whenever a node in the tree is clicked, the corresponding page is displayed in the browser window and that node is marked as the current node of the tree, and whenever a page is visited in the browser that has already been visited the corresponding node is highlighted in the tree.

Size coding on a tree node is used to indicate the number of visits to the corresponding URL. As the number of visits to a web page increases, the relative size of the corresponding tree node also increases, reflecting higher number of visits.

The domain panel displays a list of all the domains visited thus far by the user. Each domain name is a clickable link. It has a corresponding tree which can be displayed on the tree panel by clicking on the domain name in the domain panel. When a user clicks on a link in the browser window or enters a new URL and the domain corresponding to this URL does not exist, a new domain is added to the domain list and is made current. The current domain is color coded red which distinguishes it from the other domains in the domain list that are in blue.

All the frame separators are elastic, i.e. the user can adjust the size of any panel and even completely hide the two DTB panels (and let it do its job it the background). When the tree becomes big, the user can increase the size of the tree panel to get a more detailed view. This is shown in Figure 2 in which the domain names are completely hidden. The tree can also be zoomed in and out to display detail on context.

Since the main focus of our work was to organize the web histories based on domain names, we have provided some basic manipulation capabilities on the domain names. There is a search bar on the top where the user can specify any string, and DTB then displays all the domain names in the domain panel that contain the query string. For example, if the user types "cs", then all the domain names containing "cs" will be displayed. We also provided buttons to sort
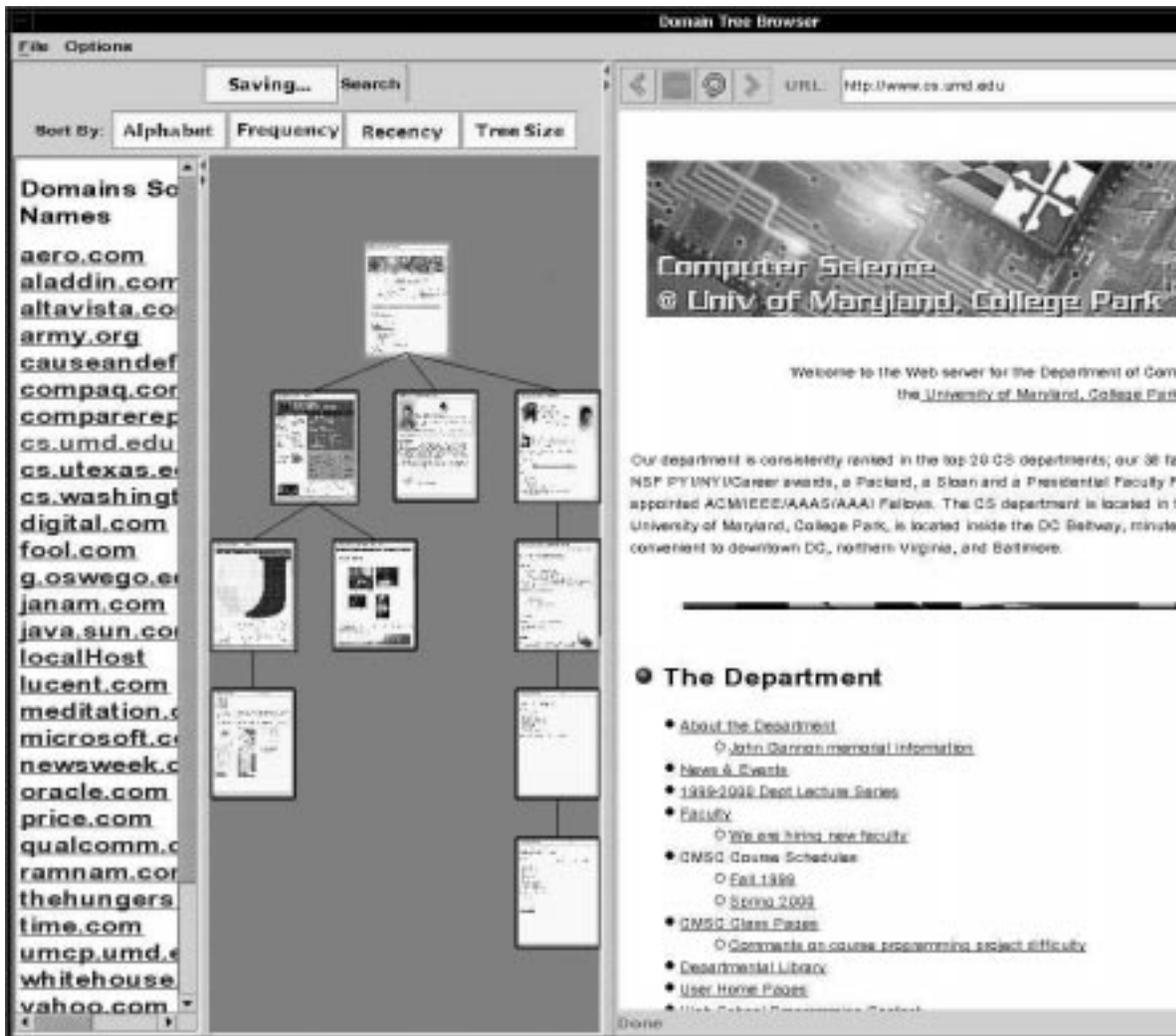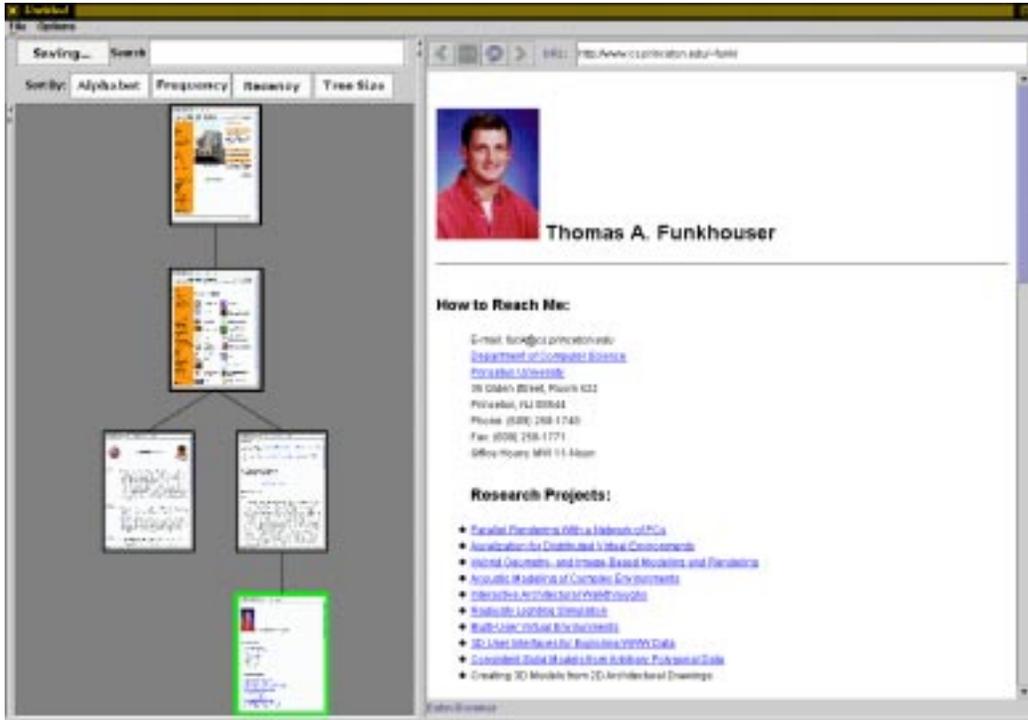
![A screen shot of the Domain Tree Browser application window titled "Domain Tree Browser" with File and Options menus. The left panel shows a "Sort By" toolbar with Alphabet, Frequency, Recency, Tree Size buttons and a "Domains Sorted by Names" list including aero.com, aladdin.com, altavista.com, army.org, causeandeffect, compaq.com, comparerep, cs.umd.edu, cs.utexas.edu, cs.washington, digital.com, fool.com, g.oswego.edu, janam.com, java.sun.com, localHost, lucent.com, meditation, microsoft.com, newsweek.com, oracle.com, price.com, qualcomm, ramnam.com, thehungers, time.com, umcp.umd, whitehouse, yahoo.com. The center panel shows a tree of thumbnail web page previews. The right panel shows a web browser with URL http://www.cs.umd.edu displaying the Computer Science @ Univ of Maryland, College Park page.]

Figure 1. A Screen Shot of Domain Tree Browser

Figure 2. Elastic windows: The domain names are hidden.

the domain names based on four criteria: alphabetically, by frequency of visits to that domain (which is the sum of the number of visits to its individual nodes), recency of visit to that domain, and the number of nodes visited in the domain.

DTB also provides the user the capability to prune a tree. The user may select the delete option under the "Options" pull-down menu, which changes the cursor to crosshair shape. If the cursor is now clicked on any node, the subtree rooted at that node is deleted. The root of a tree cannot be deleted. This feature gives the user direct control to manage the domain histories. If the user is not interested in keeping a portion of the tree then he/she can delete it. This is shown in Figure 3.

Several location probes are provided. Whenever the mouse is moved over a node in the tree, a label pops up at the cursor, displaying the URL that the node represents. When the mouse is moved over the domain sorting buttons, a label displaying the sort function is displayed at the cursor. When the user selects the delete option and moves the crosshair cursor over a node, a label is displayed at the node indicating the user that if he/she clicks the subtree rooted at the node would be deleted.

DTB provides zooming and centering. Whenever new nodes are added or deleted, the corresponding tree is resized so that the entire tree fits into the viewing area. This can be seen in Figure 3. This is animated in order to minimize loss of context to the user. The user can also manually zoom in or zoom out the tree by pressing the right mouse button and dragging the mouse to either left or right, respectively.

DTB also provides the ability to enable or disable the option of saving history. This may be useful in cases where the user temporarily does not want the histories to be recorded.

3.2 Implementation

Domain Tree Browser is implemented using Java Swing Package, and Jazz [3] which is a zoomable user interface toolkit based on Java 2D API. It uses a light weight Java Web Browser from ICEsoft [7]. The domain panel in the DTB is a *JEditorPane* enclosed in a *JScrollPane* which provides scrolling whenever the contents extend beyond the viewable area. The domain names displayed in the domain panel are actually HTML links, and we handle the *Hyper-LinkEvent*s that are generated whenever any of the domain names are clicked.

The list of visited domains is maintained using a hashtable that is separate from the browser's internal data structures. When a document is visited, the domain name of the document is looked up in the hashtable, and if it is not found, a new domain is created. A node corresponding to the document is then added to the domain's tree, if a node corresponding to that URL doesn't already exist in the tree. Two nodes are identical if their URLs are *exactly*

the same. DTB makes no attempt to determine if two different URLs reference the same document, so sometimes the same document can appear more than once in the Domain Tree Browser.

The tree panel is a *ZCanvas* (a subclass of *JComponent* in Jazz), which provides zooming and panning capabilities. To layout the hierarchy in the form of a tree, we are using Jazz's *TreeLayoutManager*. The centering and automatic zooming of the tree (on addition of new nodes) is handled using this layout manager.

The thumbnails are generated by continuously taking the screen grabs of the web browser window, until the image becomes stable, the user clicks the Stop button, or the user clicks a hyperlink and initiates loading of another page. We keep a timer that generates ticks at regular intervals of two seconds, and a screen grab is taken at every tick. The screen grabs are taken continuously because we want to obtain the best possible image, even though the user may stop loading of the current page, either by pressing the Stop button, or by going to another web page.

4 Usability Study

We conducted a usability study to determine the usefulness of DTB. Our study focussed on comparing the effectiveness of using domains to organize the visited URLs as against maintaining a single tree for all visited pages. DTB was modified so that it doesn't do any domain separation, and thus has a single tree consisting of all visited nodes. Henceforth, we will refer to this version of DTB as Single Tree Browser (STB). STB models the design of PadPrints [6] Our conjecture was that DTB would save time in returning to previously visited pages as compared to STB.

The results of the usability study only describe the qualitative outcomes of the experiments. The actual numbers have been intentionally omitted due to lack of statistical significance. Our study only tried to find out how the users might find domain based tree organization of web histories useful, in contrast with a single tree. A more detailed study would involve allowing the users to use DTB over a longer period of time and logging the features most used, the number of pages visited on an average to find a specific page, etc.

4.1 Subjects

Four subjects participated in the usability test. Two of them were graduate students in the Computer Science department. The other two were graduate students in non-engineering fields.

4.2 Training

Subjects were trained in use of STB and DTB. Subjects were already familiar with the history mechanism, bookmarks capability and the Forward and Back keys of the Netscape navigator. Training of STB and DTB included informing the subjects about visualizing web histories and telling them the difference between the two visualizations (domain based trees versus non-domain based trees). They were also informed how these differ from conventional history keeping mechanisms. For DTB, subjects were informed about the search capability that is provided. They were then instructed to visit a series of pages and revisit them using both visualizations. They were also instructed to sort the domains by different parameters, and to use the search field to locate specific domains.

4.3 Tasks

We captured the amount of time and the number of page accesses required when a page needs to be revisited. Subjects were instructed to visit the web pages of different universities in North America and specifically the web pages related to academic departments and admissions. The subjects were then asked questions that required them to visit the pages that they had already visited. Example tasks were

- Compare the tuition cost of studying at University of California, San Diego and University of Colorado, Boulder.

- Go to the web page of the history department at the University of Texas at Austin.

- Go to the web page of Saul Greenberg, who is a faculty member at a university in Canada.

For each subject, the time to answer a question and the number of pages accessed were recorded. Our conjecture was that DTB would save time in returning to the previously visited pages.

4.4 Results

The mean time to answer a question using DTB was lower than that using Single Tree Browser. The number of pages accessed to get to a previously visited page were also slightly lower with DTB. In DTB, it was observed that most of the time was spent in searching for a specific node within a tree. Since the pages visited are categorized by domain names, the tree size of each domain is relatively small and this reduces the search time as compared to STB. The users were able to get to the desired domains pretty quickly. For the task of going to the web page of Saul Greenberg, a

(a)

(b)

(c)

(d)

Figure 3. Pruning along with Zooming and Centering. (a) a screen shot of DTB. (b) two subtrees added to the center node of the tree in (a). (c) the resulting tree when the rightmost subtree of the center node is pruned. (d) the rightmost subtree of the center node is again pruned to give us the tree in (a).

faculty at a Canadian university, two of the users were not able to reduce the search space by searching for ".ca". They gave "edu" as the search field and subsequently spent most of their time finding the appropriate domain tree.

The users expressed a greater overall satisfaction using the DTB. They found the organization of history data based on tree domains to be especially useful, because that resulted in smaller, more manageable trees. The users expressed desire for an ability to search for specific nodes *within* a tree.

5   Shortcomings

One of the drawbacks of visualizing histories using domain based trees is that it doesn't depict the parent-child relationship exhibited by PadPrints [6]. In PadPrints a child node represents a web page reached from the web page of its parent node in the tree hierarchy. In DTB, when the user visits a new page in a domain *D1* by following a link in a page in domain *D2*, a node corresponding to the new page is added as the child of the *current page* in domain *D1*. However, the new page may not even be reachable from that *current page* in domain *D1*. For example, in Figure 4, the node corresponding to URL http://www.cs.umd.edu/~ben is not directly reachable from the URL http://www.cs.umd.edu. However, DTB relates the two as parent and child because the node corresponding to http://www.cs.umd.edu/~ben was added when http://www.cs.umd.edu was the current node in the domain "cs.umd.edu".

Figure 4. The parent-child relationship does not depict reachability

One way to depict this unreachibility is to encode it in the representation of the link (for example, showing the link as a dotted line).

6   Design Considerations And Future Directions

There are many interesting ways to extend the Domain Tree Browser.

As mentioned in the previous section, if a user visits a page in a domain by following a link in another domain, a relationship exists between the two domains. But the Domain Tree Browser fails to capture that relationship. If trees are used to represent the visited pages within a domain, and domain separation is done, we need to design some mechanism to reflect such a relationship.

Another issue with using tree structures is whether to display the tree *top-down*, which supports long and skinny trees or to display it *left-right*, which supports trees with a high fan out. PadPrints [6] and MosaicG [1] use left-right tree display. One design choice is to give the user an ability to select the tree layout (through a pull-down menu or a button). Another option would be to do it automatically, by fixing some thresholds, beyond which the layout of the tree toggles between the two layouts.

DTB requires a richer set of tree editing capabilities. The users may want to prune not just the subtrees, but some specific nodes. One way to delete a specific node would be to make all its children, the children of the parent of the node being deleted.

The users may also want to pick a subtree, detach it from its current parent, and place it under another node. Lets take an example. If the user first goes to the HCIL website, www.cs.umd.edu/hcil, and then goes to the CS department website - www.cs.umd.edu, the node representing the CS page will be the child of the node representing the HCIL page. But the user may want to make the CS node the parent of the HCIL node.

The users may only be interested in a portion of the tree, and may want to temporarily hide sub-trees from view by shrinking them so that they occupy very small screen area. A visual cue of the presence of a subtree could be provided to the user by marking the shrunk subtree as a circle.

For better screen space utilization, DTB could replace the links in a long chain of nodes by partially overlapping the nodes, and not displaying the links. This would enable the screen space to utilized more efficiently.

It would be useful to be able to save the histories to the disk, including domain names, and corresponding tree structures (with screen grabs) for later use. DTB could automatically upload the user's entire history from files on the disk, whenever it is restarted.

For many applications, it may be useful for the user to write some annotations on specific nodes. For example, a

user shopping for a new car on the web might visit several car pages. In such a situation, the user may want to record the key points of each car so that the user does not have to search through the entire web page whenever information on a car is subsequently needed.

Some more location probes could be incorporated in DTB. When a mouse is moved over a domain name in the domain panel, its attributes like the number of visited nodes in that domain and the time of last visit to any node in the domain could be displayed using a pop-up label.

For faster access to specific nodes that have been visited, a search capability should be provided to search for a specific node within a tree.

A capability should be provided so as to allow the user to be able to view the most recently visited *nodes*. The selected nodes (based on how many the user wants to view) could be displayed on the tree panel laid out as a grid, and clicking on any of the nodes would cause the corresponding URL to be uploaded in the browser and the corresponding page would be displayed.

The user may also want to incorporate their bookmarks in the history keeping mechanism.

7 Conclusion

We conclude that organizing URLs by domains and visualizing each visited domain appears to be an effective way to visualize histories. The usability study shows that the users took less time with the DTB browser to revisit already visited pages, and more time without DTB. They users also expressed an overall increased satisfaction while using DTB. However, this was a preliminary study and there are several issues (related to design and interface) that need to be addressed to enhance the utility of DTB.

8 Acknowledgments

References

[1] Ayers E., Stasko J. Using Graphic History in Browsing the World Wide Web. *Proceedings of the Fourth International World Wide Web Conference*, Boston, MA.(1996) *http://www.w3.org/Conferences/WWW4/Papers2/270/*

[2] Bederson B., Boltman, A. Does Animation Help Users Build Mental Maps of Spatial Information. *Proceedings of InfoViz '99*, Los Alamitos, CA., IEEE, (1999). *ftp://ftp.cs.umd.edu/pub/hcil/Reports-Abstracts-Bibliography/98-11html/98-11.html*

[3] Bederson B., Mokwa J., Good L. Jazz Website, University of Maryland, College Park. (1999). *http://www.cs.umd.edu/hcil/jazz*

[4] Catledge L., Pitkow J. Charaterizing Browsing Strategies in the World-Wide Web. *Proceedings of the Third International World Wide Web Conference*, Darmstadt, Germany.(1995) *http://www.igd.fhg.de/www/www95/proceedings/papers/80/userpatterns/UserPatterns.Paper4.formatted.html*

[5] Doemel P. WebMap - A Graphical Hypertext Navigation Tool. *Proceedings of the Second International World Wide Web Conference*, Chicago, Il.(1994) *http://www.ncsa.uiuc.edu/SDG/IT94/-Proceedings/Searching/doemel/www-fall94.htm*

[6] Hightower R.R., Ring L.T., Helfman J.I., Bederson B.B., Hollan J.D. Graphical Multiscale Web Histories: A Study of PadPrints *ACM Conference on Hypertext,* (1998). *http://www.cs.umd.edu/hcil/jazz/learn/papers*

[7] ICEsoft Website. *http://www.icesoft.com*

[8] GVU's WWW Surveying Team. "GVU's 6th WWW User Survey", Georgia Tech, Atlanta, GA., (October 1996). *http://www.cc.gatech.edu/gvu/user_surveys/survey-10-1996/*

[9] Tauscher L., Greenberg S. How People Revisit Web Pages: Empirical Findings and Implications for the Design of History Systems. *International Journal of Human Computer Studies*, Special issue on World Wide Web Usability, 47(1), p97-138, Academic Press.(1997) *http://www.cpsc.ucalgary.ca/grouplab/papers/1997/97-HowUsersRepeat.IJHCS/RevisitArticle.html*

[10] Wexelblat A. History-Based Tools for Navigation. *Proceedings of the Hawaii International Conference On System Sciences(HICSS-32)*, IEEE Computer Society Press, (January 1999).

[11] Wexelblat A., Maes P. Footprints: History-Rich Tools for Information Foraging. *Proceedings of CHI'99 Conference on Human Factors in Computing Systems*, ACM Press, (1999). *http://wex.www.media.mit.edu/people/wex/Footprints/footprints1.html*