Toward an Agileviews WWW Sitemap Kit: The Generalized Relation Browser

Benjamin Brunk and Gary Marchionini
[brunkb,march]@ils.unc.edu
Interaction Design Lab
School of Information and Library Science
University of North Carolina at Chapel Hill
December, 2000

Abstract

This chapter describes the design, development and testing of one component of a system for creating generally applicable, data-driven sitemap tools and information retrieval applications. This category of software tools are used to enhance information seeking on the web. Such tools are set up by web site administrators who want to provide visitors with alternative browsing and navigation aids. The Generalized Relation Browser, or GRB, illustrates the look-ahead strategy for web navigation within the Agileviews framework. Agileviews define control mechanisms and interfaces for overviews, previews, reviews, peripheral views, and shared views intended to help people make better decisions while seeking information on the Internet. Within this framework, the overall goal of the GRB is to help users to gain a better understanding of collections of online resources by providing them with a kind of sitemap. The GRB is the follow-on to the Federal Statistics Relation Browser prototype, which focused on the US Government Fedstats collection of web sites, providing users a more dynamic and informative browsing alternative to the static sitemap.

Keywords

Browsing; sitemap; web navigation; previews; overviews; agile views; relation browser; interface server;

Introduction

The process of finding information on the web has not changed significantly in recent years. The most expedient way to find information on the web is through the use of a search portal (e.g. Google) or categorical directory service (e.g. Yahoo). These facilities are fairly good at helping people get nearer to finding information, but they can't do everything. Searchers invariably end up browsing. A list of potentially relevant documents (pages) is generated in response to a search query, and then the searcher uses the web browser to pore over the list, page by page, jumping to potentially relevant links, while making heavy use of the "back" button to return to their search results page time and again. The results of a search may or may not contain anything useful. The only way to find out is to examine the search results and decide which ones to visit based on the meager amount of metadata provided: the URL, the page title, and usually a short text description or the context in which the keyword was found. Thus, the searcher is making a decision to follow a link based on very limited information. Visiting each search "hit" to see what is there is a very time-consuming process. Searchers often find that the information about a page in the search result proves to be inaccurate once they get to the page (Amento, 1999), which leads to frustration. Useful and relevant information is commonly overlooked, ignored, or abandoned.

In addition to the above problem, browsing hypertext has not changed much since the web became mainstream in the early 1990s—you scroll through a page and click on a link and jump to a new page. Conklin noted that there are two fundamental problems to navigating hypertext: Disorientation and cognitive overhead (1987). Disorientation is the tendency to lose one's sense of direction while navigating hyperspace. Jakob Nielsen sums up disorientation with the quote: "The jumping metaphor leaves the user with no context as to where they are in hyperspace" (Nielsen, 1990). Cognitive overhead refers to the additional effort and concentration necessary to maintain several tasks or trails at one time.

Of the many tools available to aid web browsing, none have been able to alleviate the fundamental problems associated with hypermedia such as information overload and high cognitive overhead. In attempt to lessen the impact of these ongoing problems, we have begun working on a suite of tools using the Agileviews concept. The Generalized Relation Browser, or GRB, is the first of these.

Background

Marchionini (2000a) has conceived of a framework called Agileviews. This framework approaches the design of browsing and search interfaces from a human perspective. Agileviews recognize that search is

embedded in real-life tasks. People are interested in getting work done, not in learning how to use new software tools.

Agileviews consists of six distinct components that relate to someone conducting a search or otherwise engaging in information seeking. The *primary view* is represented by the stacked rectangles at the right in Fig. 1. Overviews and previews provide information seekers an ability to look ahead before they must focus on distinct primary views. Reviews provide historical information about the current or past activity. Peripheral views provide the context for the current focus and includes any views in windows that are on the screen but not currently active. Shared views make use of the skills, knowledge, and time of other people. Marchionini, et al. (2000b) provide a detailed description of the entire Agileviews framework. Here, we will focus on two of its components, overviews and previews.

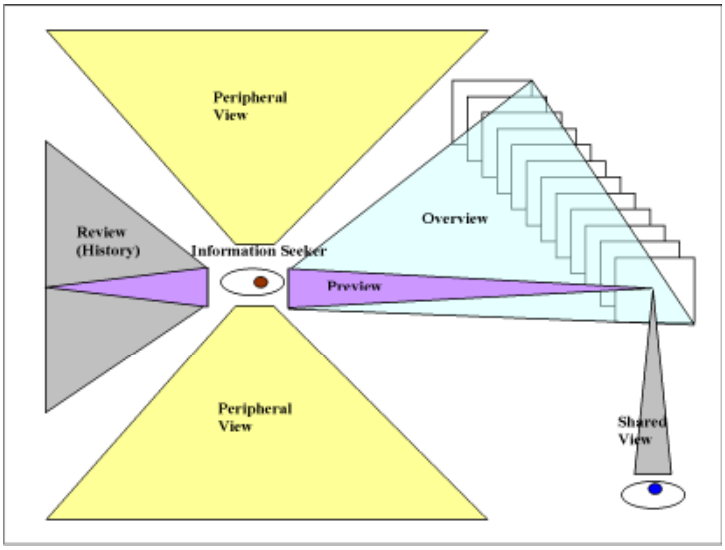

Figure 1. Agile Views Model (Marchionini, 2000b)

Overviews are used to help people find the boundaries of an information space. With an overview, they learn about what is and is not available, how information objects relate to one another, and what levels of granularity exist in these relationships. Overviews help to reduce information overload by letting people see a less-detailed view of an information space and allow them to become more comfortable with it rather than overwhelmed by it. Greene et al. (2000), note that since information objects may be considered from many levels of detail, overviews can be characterized as views which provide different perspectives on collections of those objects. Information visualization techniques have been applied to create overviews, and there are many examples in common use today. Among the first of these was the Information Visualizer suite from the Xerox PARC group (e.g., Card et al., 1991; Lamping & Rao, 1996). The Information Visualizer included the perspective wall, and the cone/cam trees. More recent examples include Lin's semantic maps (Lin, 1997), and the Human-Computer Interaction Laboratory's (HCIL) starfield displays (e.g., Shneiderman, 1998).

Whereas overviews give people views of collections of objects, *previews* focus on specific, individual information objects. The issue of granularity comes into play when deciding what is a preview and what is an overview (e.g., a web page is an object at one level but also a collection of associated objects such as texts, images, and link anchors). Previews bring users closer to finding an answer to a specific question and facilitate a probability decision about getting closer to the problem solution. Like overviews, there are many examples of previews in common use such as movie and thumbnail images on web pages. Greene et al., (2000) and Marchionini, et al (2000a) provide several more examples of previews as well as overviews.

The notion of previews and overviews applied to hypertext browsing arose from the Information Science literature in discussions addressing the problem of information overload in digital libraries. Previews and overviews are graphic or textual representations of objects of interest (Greene, 1999). Both forms add value to the original object or collection of objects by enhancing human accessibility to and understanding of the things they represent. Previews and overviews promise to be very important to the web now and in the future. They can help speed up the process of finding and making decisions about objects on the web, just as they do in the physical world. Sitemaps are an important application of the concept of previews and overviews. To further the discussion of previews and overviews, we need to examine the ways in which they are commonly manifested. The most common previews and overviews found on the web appear in the form of sitemaps.

Site maps are not actually maps at all. They come in many different forms, but most are some variation of a table of contents that shows the hierarchical structure of the pages at a given site. There are two main classes of sitemaps: those created from static text and graphics and those that are data driven.

In the former case, most of these sitemaps depict a one or two-level hierarchy view of links and are created by hand by someone managing the individual web site. This kind of sitemap is normally hand made and can only be updated by the webmaster editing a file or graphic. Examples of these types of sitemaps abound. Figure 2 shows an example of a simple html-based sitemap made by someone at the Texas Department of Economic Development, circa 1998 (TDED, 1998) showing a view similar to an index or table of contents for the entire site.

Figure 2. TDED Sitemap using basic HTML (1998)

Figure 3. NASA Sitemap (1998)

Another example is the NASA web site index (NASA, 1998) shown in Figure 3 An important difference to notice about the NASA sitemap is that although it represents a site consisting of thousands of pages, the sitemap showcases only a select subset of them. We do not know how the NASA site's designers arrived at the decision of which pages to include, we can only trust their expert judgment.

The Boeing Company sitemap in Figure 4 makes use of HTML tables and colors to help users understand how the web site is structured hierarchically without having to make use of graphics to do so. The Boeing sitemap includes textual descriptions for each link, which provides useful preview information. Unfortunately, the inclusion of so much text makes the sitemap too big to fit in the browser window, forcing users to have to scroll, preventing them from seeing the entire sitemap in a single glance.

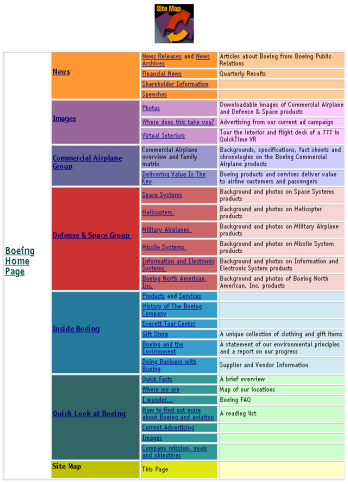

Figure 4. Boeing Corporation Sitemap

The invention of clickable image maps allowed for the next great leap forward in sitemap design. Hand-drawn network graphs, and metaphorical diagrams can be created using them. These sitemaps are effective in depicting a hierarchical structure, or laying out links spatially according to categories. Figure 5 is an example of a clickable image map created by Apple Computer Corporation for their web site, circa 1995. This sitemap shows a hierarchy of nodes surrounding a central node representing the site's main entry point. It is a concise and visually attractive drawing that summarizes the content of the Apple web site through the display of a three-level hierarchy. Like the other examples, however, it cannot be easily updated. Graphical sitemaps can not avoid the problem of limited screen space. If anything, the creators of graphical sitemaps must be even more attentive and clever in order to create a worthwhile overview of a site, adding additional time and cost to the overall site development effort.

None of the techniques described thus far work well when scaled up to hundreds or even thousands of links. Static hierarchical views are limited in that they can only show a single kind of relationship between information objects. There is another class of sitemaps that have come into existence expressly because of

the limitations of the static sitemapping techniques. These are the data-driven sitemaps. Most of these make use of Java or a scripting language because graphics play a big part in creating sitemaps, and Java provides everything necessary to create highly-accessible, event-driven, graphical applications. Java programs can be embedded in a web page and make use of multiple threads and can even run on the server side (e.g. servlets). All this opens the door to a new world of options for creating dynamically updated overviews and previews of web content that make use of novel visualizations to represent information objects. Consequently, data-driven sitemap tools represent the state of the art in agile views.

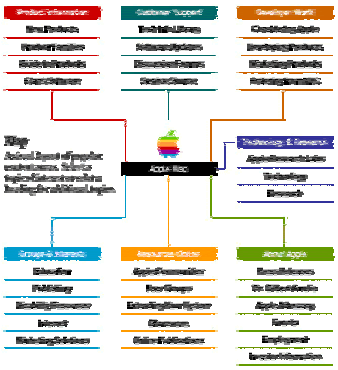

Figure 5. Early Apple Computer Sitemap (1995)

From a user interface design standpoint, the data-driven tools are more interesting as most use graphical visualization techniques and imply greater user interaction and engagement. For example (though one that is no longer commercially available), Starnine LiveIndex (Starnine, 1995) shown in Figure 6 is an example of a Java-based sitemap tool that offers an expanding tree view similar to the one in the Microsoft Windows Explorer interface. Users can expand and contract the folder nodes to reveal the pages beneath. Clicking on a page (leaf with no sub-nodes) advances the web browser to the page. A two-dimensional hierarchical view is a common visualization technique used in sitemaps. LiveIndex improves upon the static tree view method by allowing the user to expand and contract parts of the tree in a dynamic fashion. In addition to being more interactive, the major advantage of LiveIndex over a hand-drawn graphic is that its contents can be easily updated by editing an initialization file. LiveIndex is capable of displaying a broad, deep hierarchy while presenting it to users in a relatively small amount of screen space. Users are able to adjust the level of detail by expanding or contracting the folders. Since web sites don't have folders (sub-directories) or files unless one is exploring the underlying filesystem on which the pages reside, such a metaphor is somewhat out of place. A better metaphor would be to display only one node type. This way, each node represents a page and all nodes can have child nodes, in the same way that any web page can have links to sub-pages. A major limitation of this tool is that displaying a hierarchy that is very broad or very deep would be that the user has to scroll in order to view various parts of the tree, it can't all be seen at once. LiveIndex is best suited for sites of small to medium-size with a node depth of five levels or fewer. Another limitation is that the text labels on each node provide only minimal preview or overview information about the pages they represent. It is only marginally better than a sitemap made using a clickable image map like the ones discussed previously.

An example of the expanding outline technique combined with added overview features is WebTOC (Nation, 1997). WebTOC (Figure 7) is a Java applet that provides a table of contents inside an HTML page on the left hand side of the browser window. The WebTOC package was originally written to enable researchers to browse processed and unprocessed (e.g. no indexing or meta-data) collections related to the Library of Congress National Digital Library Program. WebTOC goes beyond the functionality provided by LiveIndex and is actually two applets: one is a crawler, the other is a viewer. The crawler is given a seed URL and it can generate data by following the HTML links starting from the main page, or it can use the underlying file system structure, generating a hierarchical representation of the documents local to the site. The Viewer then displays the gathered information in a tree view, or table of contents. In addition to the text, each local document is represented by a colored line, its

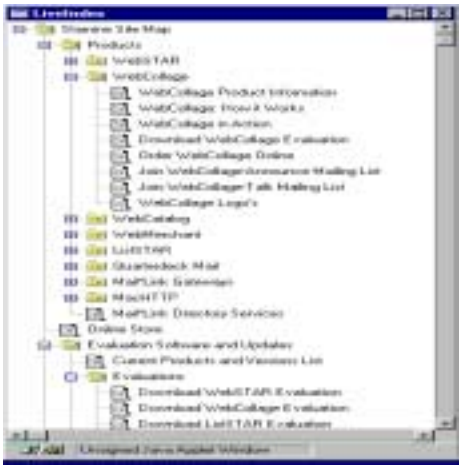

Figure 6. Starnine LiveIndex (Starnine, 1995)

length corresponding to the size of the file. The color of the line represents the type of file (e.g. text, image, audio). If a document contains links to other documents, the lines representing the documents it includes can be collapsed into a thicker "size bar" that shows the total number of the documents it references. Each size bar has a shadow under it. The size of the shadow indicates the number of items subordinate to the current file, giving users a visual cue to help them distinguish quickly between items that have a few subordinate links and items that have many links. As an alternative, a bar representing the total number of items can be shown. The user has the option of viewing just the colored lines without the text descriptions. This more compact visual representation makes size comparisons easier (Nation, 1997). The added information gives users a very good overview of a web site. WebTOC does take up a large amount of screen space, but instead of appearing in a frame, it is possible to make it appear in a separate popup window that can be minimized when not needed. WebTOC can scale to handle a large amount of content; although deep hierarchies require the user to perform a lot of scrolling.

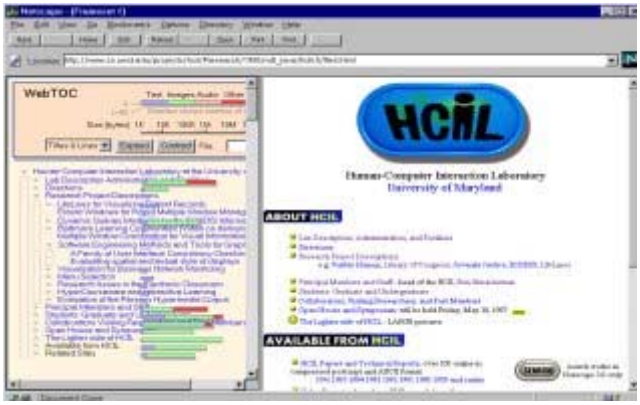

Figure 7. WebTOC Viewer (Nation, 1997)

As we have seen, the goal of data-driven sitemaps is to provide their users with better overviews and previews of sites. The data-driven tools we have just seen are of limited use because they take up so much screen space and rely heavily on textual descriptions. There is another sub-class of data-driven tools that are more graphical and object-oriented that have grown out of research projects in the field of Information Visualization.

Information visualization describes concepts for visual user interfaces in text retrieval, including tools and methods for representing abstractions and surrogates for textual information, which is stored in databases (Däßler, 1998). Information visualization combines aspects of scientific visualization, human-computer interaction, data mining, imaging, and graphics. Research in this area is currently very popular due to the increasing need to find methods for displaying larger and larger amounts of text and other data in efficient ways.

Among the pioneers in the Information Visualization field, George Furnas proposed a set of criteria for designing and evaluating representations of information structures in support of traversal and navigation (Furnas, 1997). One of his early works extends the idea of balancing local detail within a global context and merging it into a single view by using what he termed a *fisheye view* (Furnas, 1986). The fisheye view is analogous to a wide-angle camera lens. Furnas' idea was to show "local" detail in full, and then provide context to that detail all around it, but in successively less detail. He described a generalized degree of interest function, where the interest value of a node in a graph is a function of both its a priori importance and its distance from the user's current focus. There are three methods for creating a fisheye view: Graphical distortion of the context areas, partial views through filtering, or by using smaller or simpler abstractions such as icons in the context areas. This technique could be adapted to provide a global overview or a preview of a web site, since fisheye views are capable of representing large, non-hierarchical information spaces.

Other early visualization work emerged from a trio of individuals, Card, Robertson and Mackinlay, at the Xerox Palo Alto Research Center (PARC). Cone trees (Figure 8, Robertson, et al., 1991) Perspective Wall (Mackinlay, et al., 1991) and the 3D Information Visualizer Workspace (Card, et al., 1991) were important early visualization examples. The types of visualizations they created can represent various types of associations between information objects focusing mainly on hierarchical relationships. These early prototypes include many of the features desirable for creating interactive, graphical sitemap tools, and their ideas have been adapted for use on the World-Wide Web.

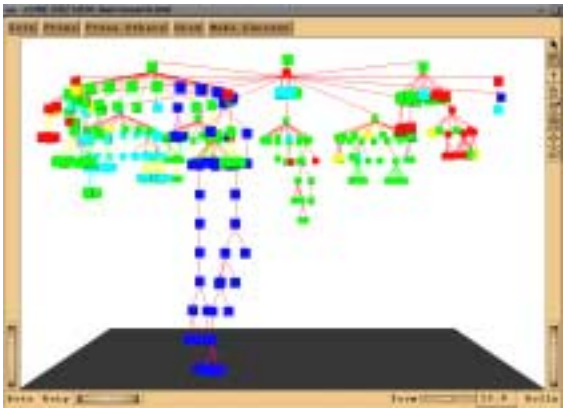

Figure 8. Cone Tree Visualization (Robertson, 1991)

Hyperbolic trees (Lamping & Rao, 1996) allow users to manipulate large hierarchical structures to see overall structure and find specific pages. Their great advantage is that although they achieve the same functionality as cone trees, but can display up to 10 times as many nodes as a conventional 2D browser in the same amount of space. Hyperbolic tree views are more compact because they make use of a non-euclidean geometry that enable large hierarchies to be laid out as though they were mapped on the surface of a sphere. By using animation and dynamic focus, users can manage extremely large information spaces with ease. Hyperbolic trees have an inherent dependence on hierarchical relationships in the same manner that cone trees do. Also like cone trees, the colored nodes that represent individual pages often do not provide adequate preview information, although new designs that use mouseovers to pop up additional preview information are now available.

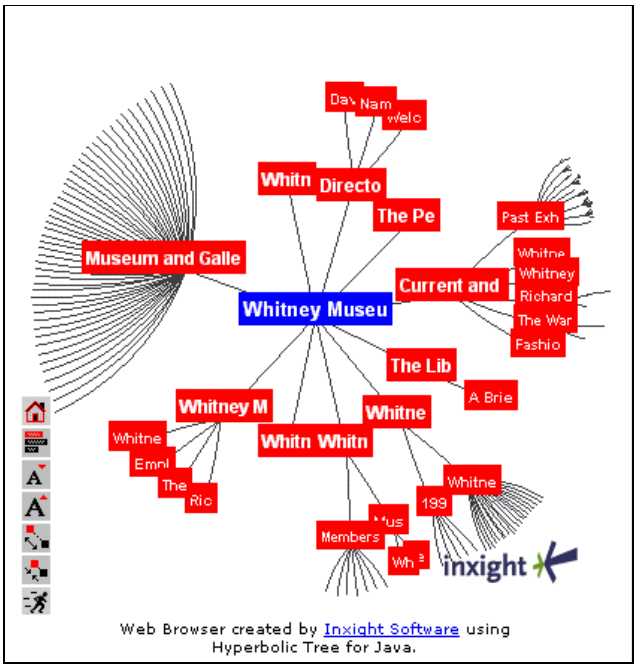

Figure 9. InXight SiteLens (now called Tree Studio) Hyperbolic Tree Tool (InXight, 1998)

Inxight Software (a Xerox PARC spin-off) has recently released a hyperbolic tree visualization product called SiteLens (InXight, 1998). Each node in the SiteLens represents an individual page at the site. Users can rotate a node to the center of the view by clicking on it, or they can drag the whole tree around by clicking on a node and holding the mouse button down. Double-clicking on a node advances the user's web browser to the page represented by the node. Like WebTOC, this applet gets its data from a separate web-crawler applet that crawls a web site and creates a data file with the hierarchy information, page titles, and URLs.

Many relationships within a complex site are not hierarchical. Instead, they link distinct instances of different attribute sets, which are collections of content attributes. These types of attributes are typically handled by database management system tables that support common user queries for certain relationships. In site maps, the aim is to allow users to explore arbitrary relationships between attributes without having to pose formal queries. As previously noted, most site maps only provide one specific type of relationship exploration (e.g. hierarchy). But many other types of relationships exist and can be useful to users. Graphical maps show spatial relationships, timelines show temporal relationships, and entity relationship diagrams show relationships between distinct entity sets contained in tables.

The Treemap visualization is one that can show relationships that exist between attributes in a collection of information objects (Shneiderman, 1992). This approach uses a 2D space-filling algorithm in which each node is a rectangle whose area is proportional to some attribute such as node size. It's original application was as a file manager for a Macintosh computer. The files were shown as leaf nodes and the subdirectories were interior nodes. Later work added color-coding and node proximity as well as some esoteric features such as zooming, sound (as a redundant or independent code, for example, larger files had a lower pitched sound), hue/saturation control, many border variations, and labeling control. The initial prototype was capable of visualizing 5,000 node hierarchies in a relatively compact amount of screen space.

Other applications for space-filling virtual layouts have surfaced. Lin has designed a program entitled Visual SiteMap that uses neural network techniques to produce semantic maps. Using a web crawler, SiteMap traverses every link of the web site, collects statistical data, and indexes all the words and pages of the site. SiteMap then converts each page of the site into a vector based on the statistical data and the indexing, using these vectors to train a neural network. The trained neural network creates an organized map: subject areas are identified and labeled; their sizes and locations are determined by relationships among the subjects and by their occurrence and co-occurrence frequencies. Links are clustered and located within their respective subject areas, represented by colored dots (Lin, 1999). It is important to note that semantic maps are not dependent on hierarchical relationships, but map proximity and size onto terminological similarities.

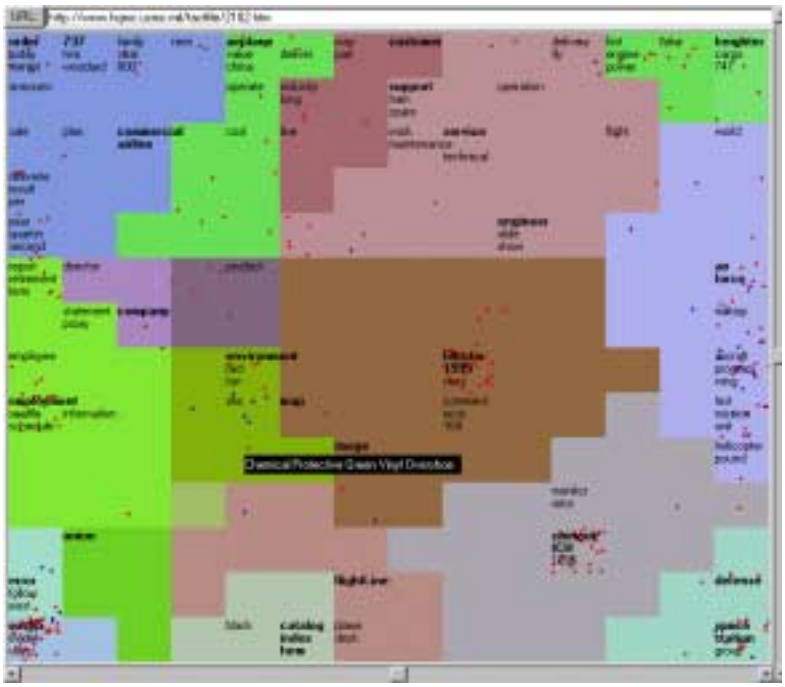

Figure 10. Visual SiteMap, an example of aTreemap (Lin, 1999)

There are, of course, other interesting examples of visualizations than we have room to discuss here. The Cybergeography web site (www.cybergeography.org) is a good online resource that provides more discussion about some of the examples discussed here, as well as many others that haven't been mentioned. The creator of Cybergeography has also written a book entitled *Mapping Cyberspace* (Dodge & Kitchin, 2000). Researchers have experimented with many other techniques for generating overviews and previews for web sites and multimedia collections. Robertson et al. (1997) used thumbnails of pages from different websites for visual bookmarking; Marchionini et al. (1997) used keyframes of video clips to give users overviews of multimedia databases; and Brunk et al. are experimenting with animated thumbnails as sitemaps (Brunk, 1999). In the WWW, relationships, other than hierarchical, cannot be shown as direct links. In a site map, it would be advantageous to be able to explore relations among various sets of non-hierarchical attributes without literally following many links. Links are expensive because they require cognitive shifts and take time to execute. A goal of the work reported here was to investigate techniques that would help people see the relationships among topics and other attribute sets of an information space such as the types (file formats) of data available, geographic origin and coverage of the data, and time period coverage of the data, all in one place.

In all cases, people must understand the control mechanisms (e.g., mouse actions) for exploring, extracting, and using information. In the human vision sense, we simply refocus our eyes to change distance or move our head to change direction. We refer to artifacts (e.g., notes) to peer back and consult our memories to make inferences about the future. Human-computer interfaces are not so natural and easily controllable. Linking pop-up windows and other focus-changing actions to mouseover, mousedown, and mouseup actions is the current best control mechanism for managing views. Although eye gaze, gesture, voice commands, and other mechanisms have been proposed, mouse events are the most pervasive solution today.

Many of the visualizations and interfaces discussed here are novel and can not be easily understood by the majority of users. Any new technology takes time for people to fully assimilate. As we began designing our own sitemap interface suitable for use on government web sites, we had to consider the broad spectrum of users that would encounter it. Previous investigations of user characteristics and their needs for statistical information (Hert & Marchionini, 1998) demonstrate that government statistical web sites must serve the entire range of citizens from school children to seasoned researchers with the broadest range of experiences and statistical needs. Thus, we needed something that would be useful to expert users, while not overwhelming the novices. Bandwidth and screen resolution were also key issues affecting our design decisions. We needed to utilize common user interface components such as buttons, tabbed panels, and table views, but then incorporate some novel features such as mouse-overs and graph bars, to come up with a useful alternative for overviewing and browsing the information available via the Fedstats portal.

The Fedstats Relation Browser

The Fedstats web site provides access to the 200 websites and thousands of web pages containing statistical reports, datasets, tables, and other statistical elements produced by 70 federal government agencies. This site (www.fedstats.gov) acts as a portal, directing people to the bulk of statistics collected and disseminated by the US Government. Fedstats offers a static HTML-based sitemap as well as a search facility as entry points to this vast array of statistical information. Over the past two years, we have been working with the Fedstats team to develop alternative access mechanisms. The first solution to come from this work was an exploration tool called the Relation Browser (RB). The overall concept of the RB was to provide the user with a mechanism to examine pair-wise relationships among attributes such as topic (economics, crime, education, etc.), data type (report, table, graph, downloadable dataset, multimedia, etc.),

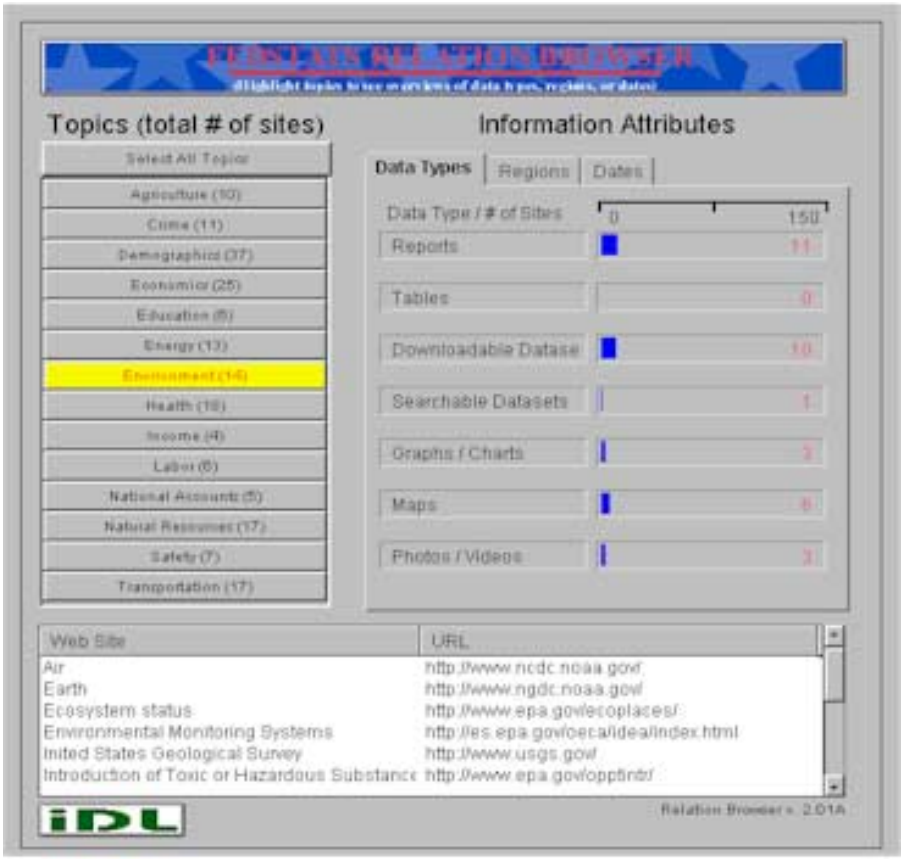

Figure 11. Fedstats Relation Browser (version used in field test)

region (international, national, region, state, substate), and time coverage period. Because the alternative site map allows people to explore pair-wise sets of relationships, the prototype was called the relation browser. The design of the RB interface aimed to help people understand the relationships among various attributes of the database. It facilitates a probability decision at the macro level about what strategies and tactics might be useful. In Figure 11, the RB interface shows a view of the Fedstats collection when the user has rolled the mouse over the environment topic. Note that the number of websites that have tables, reports, downloadable datasets and other information types are shown as blue bars and numerically so that user gains a sense of volume and type of data for this topic. The website titles and URLs appear below to provide rapid access to those websites. If the user changes the mouse position to another topic, the display instantly updates to show the corresponding data for that topic. By exploring the topics without committing to a mouse click, the user is able to gain a quick overview of the range of data in the 200 websites provided by these government agencies. Users may also choose to explore topics by geographic region or time period. We are currently working to give additional linkages to the data types in the result list to give users even more detailed looks ahead before they commit to a click. We have conducted several iterations of usability tests with the RB as well as a field test of the tool accessible at the Fedstats website. See Marchionini et al. (2000) for details of the system and testing.

Design Process

There are several ways to approach the process of designing a data-driven sitemap tool. Our goal was to improve the quality of the site navigation experience through the employment of a novel alternative to the existing sitemap while leveraging the existing knowledge and skills of the diverse user base of the Fedstats portal. One approach is to provide separate web sites for different categories of users. Many sites with small and distinct user groups are able to do this. They provide highly-specialized interfaces for each of their most common types of visitors (i.e. children, students, teachers, researchers). Along these same lines, a second approach is to perform a user needs assessment and create user models in order to understand the most common user needs and tasks. Once that is accomplished, specialized interfaces or explicit entry points to the site are created to better serve those specific user models, with the provision of a generic entry point for other cases. Both of these techniques require users to self-categorize themselves into roles that may not actually suit them.

A third approach is to design interfaces that adapt to individual users' characteristics. The common way to do this is illustrated in the many web sites that provide a "my" feature, such as mycnn.com or my.yahoo.com. These sites allow individual users to customize their browsing experience and save their preferences for later visits. The ultimate goal of this approach, and the one that the user modeling community has been seeking, is to create interfaces that adapt to users automatically. So far, however, such interfaces have not met with broad approval from users (Brusilovsky & Milosavljevic, 1998).

Our design technique follows yet another approach to this general design challenge. This approach is to create interfaces that support flexible interaction so that people can control their own paths towards meeting their information needs. In this approach, user actions are tightly coupled to results, creating immediate feedback, which allows users to quickly gain an understanding of what information is available and how to get to it. The Relation Browser interfaces gives users a way to explore the relationships among the data stored in 196 different web sites without requiring them to access any of those sites. They are thus able to learn which paths are most likely to suit their information needs, at which point they can then go on to find the needed information in a straightforward manner.

The first task in the design process was to gain a better understanding of the information space. Specifically, we needed to find some common attributes of the information objects on nearly 200 web independent web sites. Our efforts were hampered by the fact that nearly all these web sites are independently administered and adhere to varying standards for presenting their information. Based on our studies of user needs and meetings with the citizen-support staff at the Bureau of Labor Statistics and the members of the Fedstats steering committee, we learned that novice users of statistical web sites are task and topic oriented, rather than agency or data oriented. The Fedstats site uses 14 topical descriptors to organize information provided by 70 different federal agencies. We decided to adopt those same 14 attributes as our "main" or "pivot" attribute for the interface. As we examined the many web sites, we

found that other useful sets of attributes fell into three distinct categories: geographic coverage (e.g. international, national, etc.), time coverage (e.g. the years the data were gathered) and type of data (e.g. graph, chart, map, etc.) Upon selecting these sets attributes, we then further examined each web site to fill in the specific details for them. A DBMS was used to manage this growing collection of metadata.

The next step was to develop a paper mockup for our interface. This mockup was created based on an analysis of which user control mechanisms might be best for today's web environment and for our extremely heterogeneous user base. Our mockup incorporated the attributes mentioned above and specified that when a user placed the mouse cursor over one of 14 buttons (each representing one of the topical descriptors mentioned previously) it would trigger other buttons representing the data type attributes to change color. Thus, as users mouse-over a specific topic, the interface would show which data types were available for that topic (e.g. if the "health" button was selected, buttons for "graphs" and "downloadable datasets" might become highlighted).

After reviewing the attributes and respective values with the Fedstats team, a Java applet with an interface based on the paper mockup was coded. Our goal in this first prototype was to instantiate our ideas in a usable interface that would directly demonstrate the dynamic interaction between the list of topics and the attribute set of data types, which was difficult to illustrate with the paper mockup. The BLS and Fedstats teams were favorably impressed with the design. Based on the initial reactions to the paper mockup, we proceeded to add another feature so that users could immediately obtain a list of which web sites fell under which topics in the database. With this additional feature, we directly tied the overview with navigation. Users could now look ahead to each site, by way of its topical categorization, and simultaneously know the formats of the data available at each site without actually having to go to each one. In addition, after lengthy discussion, five region names (values for the geographic region attribute) and seven date ranges were agreed upon (values for the date attribute).

As we tried out different ideas and the prototype evolved, we decided that linking topic and type of data was the most important type of relationship that people would use. We felt it would be useful to give users some indication of the size or scope of information available for each topic-data type relationship. This is a difficult challenge in the WWW for two reasons. First, boundaries are difficult to establish. Where does a website end? All internal pages? The entire underlying database or only the set returned for a query? Second, what metrics should be used for site volume? Bits are misleading for multimedia data and impossible for streaming data and on-the-fly computations. We chose to address this issue with a visual approximation for the number of websites exhibiting the various attribute values. This was intended to give users a sense of how many chunks of data they will have to examine if they continue down a particular query path. In addition to the mouse-over highlighting that showed relationships between topics and data types, a value bar showing an estimate of the number of websites that had such data was used to indicate the volume of information available for these relationships. While not an ideal solution, it does serve the goal of the look-ahead strategy.

Implementation Challenges

Some of the lessons that we learned during the implementation of the Relation Browser prototypes illustrate the general problems encountered during the development of web-based, web-delivered software. Java is an incredibly powerful and rich programming language, but it does have limitations. The applet's functionality was molded as much by our design goals as it was by the limitations and trade-offs required by the technology. The foremost example of which was the requirement that the interface respond to mouse roll-over events and update the interface in real time. It was difficult to implement such a feature because getting interface components to update their views in response to a real-time event triggered by some other component(s) requires processing and network overhead. The advanced event model in Java made our job easier, but we still faced the problem of actually getting the data to the applet and classifying it by topic so that the graph bar values and the bottom results table could update. The biggest dilemma of the mouseover was that if a user rolls the mouse around quickly, the other views could not change fast enough. Our initial design queried an ASCII file, later versions sent a query to the database each time the mouse pointer entered the area of a topic button. Those solutions worked fine if the server happened to be located on the same host as the client; any network latency or server delay introduced too much lag. We

faced the choices of either eliminating the highly desirable and novel mouseover feature, or pre-caching the data on the client side, thereby sacrificing scalability. We chose the latter option because scalability was already hampered by the limitations of screen real-estate. Now, as the applet is loading, before the complete interface appears, all the data is transferred to the client and stored as objects in memory. It can then be manipulated in real time when the mouse rolls over or selects topic buttons. The results of the field study indicate that users were annoyed by the long load time of the applet. It was an lamentable trade-off to have to make.

Another implementation issue relating to the topic buttons of the first prototype is that they lacked the multiple-select feature, choices were limited to select one, select all, or select none. Results of the first usability study (see below) indicated that users wanted a better way to aggregate results. Giving users greater control over button group selections required additional software complexity. The upside was that giving users finer control over the topic buttons let them manage the results list more satisfactorily.

The model of serving software across the Internet is not trivial. Screen real-estate proved to be significant hurdle. We wanted to have a rich information display and have it fit within the confines of a normally-sized (i.e. not full screen) browser window. Originally, the vertical list of topic buttons in the prototype interface of figure 12 was not scrollable because the whole concept of the interface was to give people a concise overview without having to do a lot of mouse-clicking. Everything was fine if the screen has the resolution to display all 14 topic buttons. Unfortunately, we found that some configurations cut off the topic list and some of the buttons were inaccessible. The variability added by different operating systems, different screen resolutions, and different browsers forced us to add a scroll bar to ensure that users could always access all the topics.

Despite our best efforts, the applet's user interface developed some layout and font-size flaws when hosted on various different operating systems. The Java slogan "compile once, run everywhere," is misleading. There were no show-stopping errors, but we did have to debug the interface on each platform we wanted it to run on. However frustrating at times, this was still preferable to porting C or C++ code and recompiling.

We had hoped that we could make use of the Java Beans development model and modularize the various parts of the interface so that features could be added or removed with less programming effort. This goal ended up conflicting with our need to have good mouseover performance. Creating bean components adds processing overhead and extra code to the system. Bean components also take more time to write and debug. The benefit is that once this work is done, reusing your components in other projects is as easy as dragging and dropping them into your interface, they require little or no new programming effort (unless they need to be modified). Trying to make all the components of the system into beans was disappointing.

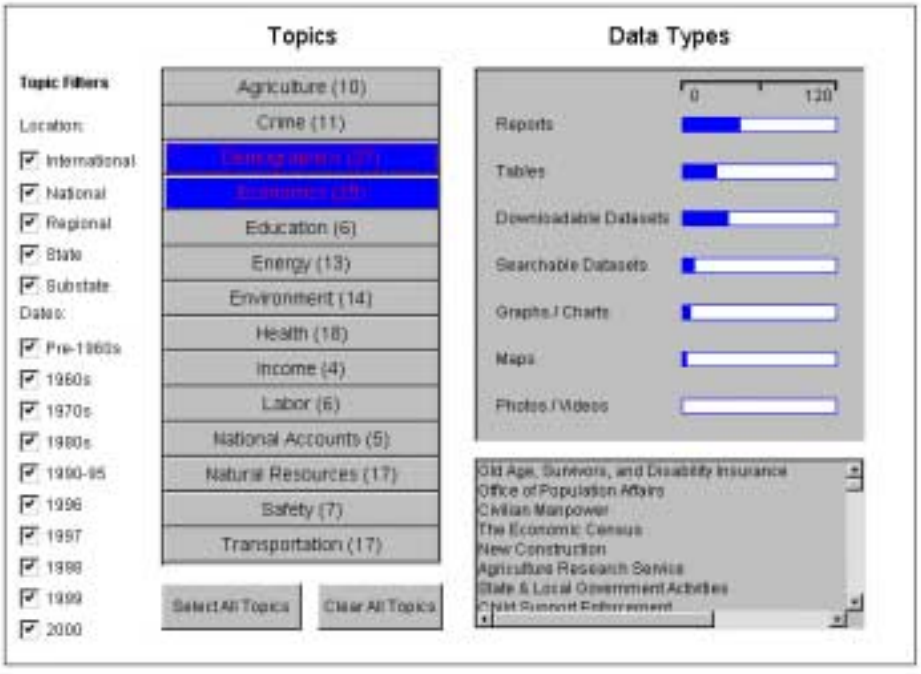

Figure 12. Early RB Prototype Interface (used in first usability study)

As code development progressed, we ran into overhead and data-encapsulation problems brought on by the bean architecture, leading us to decide to scale back our bean requirements. In the end, only two of the interface components are still "beans"—the topic buttons and the graphbars. We still feel that the Java Beans approach is worthwhile, especially if a visual editing environment is used, but introduces performance trade-offs that must be weighed against the benefits of designing highly-encapsulated, reusable components. Justifying which components of an overall project are worth making into reusable beans requires planning and testing, as well as longer development cycles.

Usability Study

To assess our first prototype, a usability test was conducted with nine subjects on the version of the interface shown in figure 12. The main goal was to determine how easy the relation browser was to use and how well it provided an overview of the Fedstats site and associated sites. Because exploration is difficult to assess empirically, a set of search tasks for the existing site map and the relation browser were devised to give test subjects some exploration guidance. In hindsight, this decision clearly biased users toward using the relation browser as a search tool rather than as an overview explorer. Testing took place in the Bureau of Labor Statistics Usability Laboratory in Washington, DC. The lab facility has two video cameras, one mounted in the ceiling to capture keyboard and mouse activity, and one mounted in the wall to capture users' facial expressions. There is a microphone to capture audio, and a scan converter to record the computer screen. All four signals are combined through a video mixer to produce a three window videotape of each session.

To focus on the site map and relation browser features, we selected subjects who were experienced WWW users. Six subjects were selected from a subject pool developed through a newspaper ad. Each subject from this group was paid $25 to participate in the study. The other three subjects were employees of the Bureau of Labor Statistics who were unpaid volunteers. Six subjects were male, three were female, and all reported using the WWW at least once a week.

The results of the initial usability study are detailed in *Look Before you Click: A Relation Browser for Federal Statistics Web Sites* (Marchionini, et al., 2000). Subjects had mixed success in using the Fedstats site map and Relation Browser to complete the six search tasks. One subject successfully completed all six tasks, and two subjects successfully completed no tasks. Subjects had equal success in the tasks using the Fedstats site map and the Relation Browser. Domain knowledge was clearly a factor in finding information. One subject who was a medical student was able to find the health care cost value quickly, and noted in the debriefing that navigation was skillful "because I was knowledgeable about terms." Another subject who worked with federal statistics regularly did not know that the CPI is produced by the Labor Department, but was able to use highly specialized knowledge about educational data to find education costs (searching for data on a specific city, knowing that the result would be compared to the national average).

Revised Prototype

Based on the user testing, a number of changes were made to the prototype. Although subjects said they liked the filters, few were able to really take advantage of them in doing their search tasks--they liked the idea of filters for searching but the relation browser is meant to give an overview of the site and become a launching point for search as a secondary goal. In the revision, the filters were treated just as the data types and a tab mechanism added to allow users to relate topics to either data types, regions, or time. This revision was more in line with the original design. In addition to the major change of making the date and region attribute sets tab-selectble alternatives to data type (rather than filters). In addition, a few other, minor revisions were made, resulting in the interface shown in Figure 11.

Field Test

After the first revision of the prototype, a link to the tool was added to the Fedstats website. Because the RB is a Java applet, only web browser versions that support Java could access the tool. In order to get public feedback for the tool, an online questionnaire was created and Office of Management and Budget

(OMB) clearance to use the questionnaire was sought. The questionnaire aimed to determine whether the RB provided users with an overview of the available federal statistics, general impressions about the usefulness of the tool, and elicited information about the settings in which users were accessing the site. The questionnaire contained 10 Likert-scaled questions, one checklist question, and two open-ended questions. Detailed results are discussed in Marchionini (2000).

Transaction logs from the web server hosting the RB tool provided additional details for our analysis. A total of 34956 requests were logged from November 12, 1999 to May 3, 2000. Because the logs consist of the applet request and three associated file requests but no records of what users actually examined while they used the applet or what sites they eventually clicked on, no attempts to do session analyses were made. One indicator of usage is that 7697 unique IP numbers were included in the requests. Because Internet Service Providers (ISPs) dynamically assign IP addresses (and individuals may use different machines), we cannot conclude that there were 7697 unique users of the RB over this time, but our approximation is that the RB was used by perhaps 7000 different users.

A total of 74 responses to the questionnaire were received during the field trial. Using the 7697 unique IP addresses as a conservative base, this represents less than a 1% response rate. These respondents voluntarily chose to complete the questionnaire and had already chosen to use an experimental/optional tool from the Fedstats site, so they cannot be considered a statistically representative sample. For the purposes of judging the potential usefulness and usability of a new interface technique, they served this research well by giving general impressions and specific reactions to the design and possible deployment.

The field test online questionnaire yielded valuable feedback on the efficacy of the Relation Browser. With the veil of anonymity, some who answered the questionnaire offered very candid criticisms of our prototype. Responses ranged from “This thing totally stinks,” to various suggestions for improving the interface, to “I wouldn't change it at all, but please make it a permanent fixture [of the Fedstats site].”

In general, the responses and commentary were positive and lent support for the RB as an alternative tool for the Fedstats site. The most common complaint was with the amount of time the applet takes to load on low-bandwidth connections. The design trade-off involving the pre-caching of data to support the fast mouseover functionality is at the root of the problem. We continue to seek an effective compromise between front-loading system overhead that results in relatively long download times and the highly desirable response of the interface during runtime that requires local data.

The experience of mounting and testing the site within the development group and in the Fedstats group was itself valuable as we learned how to define goals, explain our mission, and reflect on the possibilities for the tool and what would be necessary to deploy it as a permanent addition to the Fedstats interface repertoire.

Second Phase of Usability Study

Nine more subjects were recruited to participate in the second phase of the usability study. This time, five female and four male subjects were recruited. All were experienced professionals: four were librarians, three were news analysts/journalists, and two were research analysts. These subjects all had substantial WWW experience and were paid a $25 stipend for participation. Six of the subjects reported regularly using BLS and other statistical sites while the other three reported occasional usage of BLS and other statistical sites. Only one subject reported using Fedstats, three had looked at Fedstats but did not use it, and five had never looked at or used Fedstats before the testing.

The same testing protocol as in the previous year’s study was used with minor changes to reflect the revised tool (e.g., since there were no filters, no questions were asked about filters; since a help function was added, reactions to the help were sought, and since attribute values for specific websites were added in the results windows, reactions to this addition were sought). The same six tasks were used and subjects did three with the Fedstats sitemap and three with the RB. Sessions lasted 40-60 minutes and the same testing protocols were followed.

Detailed results of the second usability test can be found in *From Overviews to Previews to Answers: Integrated Interfaces for Federal Statistics* (Marchionini, 2000). Participant reactions to the Relation Browser ran the gambit from delight to confusion but were overall quite positive about the possibilities of using it in their work. Several noted that it would be helpful in one or more ways. They saw the possibilities for gaining overviews of federal statistical sites from several perspectives. First, some noted that the tool set up a process for understanding what is available. One noted that "it funnels you to information." Another said: "Get good jumping off point to broad overview of categories that are available. I can see what kind of websites are available." Second, one participant said it revealed relationships among the data: "To enable the searcher to understand the available websites on different subjects and to connect—to see the correlations between different subjects and relevant government agencies. To give the searcher understanding of the regions covered, types of data and time periods covered." Third, one participant noted that more information was provided: "This allows you to drill down into topics and go in to see data type, dates, etc. Gives you a lot more information, more robust." Fourth, half the participants said something about the forms of information (attributes) as beneficial features of the RB. "This is very good for getting an overview…it tells you what is included. As a researcher, I'd like this because it tells me how the data are available, plus this gives the URLs…this is a much more direct way." Another said: "The fact that a lot of this information is coming from a lot of different websites…its pulling from a lot of different sites…the way this is organized is really phenomenal…it's breaking it down by the way people really look for this information…by date, region, even though I don't really understand everything, I know about tables…downloadable datasets, I'm sure..would be a spreadsheet or something…get the number of websites. This really helps me zero in on topics…I have a much bigger overview of what is available, I'm not just blindly going down this road." Another said: "Well, I like how it gives data date and region." These participants saw several ways that the RB can provide an overview of data.

Although these comments illustrate excellent promise for the RB, half the participants made some comment about needing some time to get used to using it. One participant suggested that reporters without Internet experience would have difficulty, noting: "Oriented more to the expert." Three other participants noted: "I like it…but I'm not sure it would be used a lot by the novice." "[A] little more explanation, help." "Useful but hard for me…tricky….If I used it a lot I'd find it very useful….If I was required to find statistical data one or two times a day I'm sure I'd use it."

There were several specific suggestions for improvement. One participant complained about the terminology: "Things that aren't clear like 'information attributes' has more to do with personality like nice hair." Two subjects questioned the value of the graphical bars showing how many web sites had specific data types (redundant since the numeric values were there). One noted that the fonts were hard to read. Two respondents were bothered by the amount of screen activity. One said: "There is so much happening…when roll the mouse over, there is so much going on, roll here, something happens, go there, something else happens. Just too much going on." Also, three participants specifically mentioned that they would like to have a keyword search capability. The RB was intended to be integrated into the Fedstats site as a modular component. The sitemap and keyword search mechanism both constitute separate components. Our testing focused specifically on usability of the RB, not the overall site, so the keyword search facility Fedstats offers was not accessible during the test.

In sum, participants liked the RB but eight of the nine subjects noted that the site map was more familiar and easier to use. Subjects preferred different tools for different tasks: the RB was preferred as an *overview tool* and the site map was preferred for *finding specific information*. They noted that the RB gave more information, most commonly citing the data type, region, and date attributes and the URLs as examples of information given by the RB but not the site map. These results support the use of the RB as an overview tool that could be used as an adjunct to traditional tools such as the site map, A-Z list, agency list and keyword search tools. The additional information provided by the RB gives people new views of federal statistics but these new views come with associated costs of processing and understanding. When understood as an alternative to existing tools, rather than replacement, the RB was a welcome addition for many of the subjects.

Generalizing the Prototype

Our participation in the design process could have ended when the Relation Browser was successfully employed on the Fedstats portal. The Fedstats and BLS teams were to use the ideas and prototype code to engineer their own in-house, fully documented version of the Relation Browser, based on what they learned from our prototype. That project is ongoing.

We believed that the Relation Browser interface could be used to create overviews in any setting where a small number of attribute sets (e.g. 3-10) that each have a finite number of attribute values (preferably less than 15) could be identified. To instantiate this belief, we have created a generic tool with an interface nearly identical to the Relation Browser that allows graphical exploration of collections. It is called the Generalized Relation Browser. The updated version has had significant alterations made to its back-end processing capabilities in order to broaden its applicability and reduce the overhead of managing it. During this upgrade, the user interface was rewritten in Java 2 and its AWT-based interface was completely redone using the more versatile Swing components. Many of the remaining user-interface problems from the earlier usability studies could finally be appropriately addressed. Unlike the prototype version that gets its data from an ASCII file on the server, this version interacts directly with an open-source DBMS (MySQL, though it could easily be coupled to any DBMS system for which there is a Java JDBC bridge driver written), easing the task of managing the system. By filling out the appropriate fields of the applet tags embedded in a web page and accessing different database tables, one copy of the GRB can simultaneously provide overviews of dozens of unrelated collections on separate web pages.

Class Project

With the latest redesign finished, the stage was set to test the effectiveness of the generalized tool. Rather than conduct yet another usability study, since we felt the user interface for the tool was quite refined by now, we elected to test it in a different fashion. As part of a graduate seminar course in advanced user interface design, a group of graduate students were asked to use the GRB to create an overview for their own collections. In the fall of 2000, students taking an HCI seminar at UNC each used it to create an overview of their own databases (refer to http://squash.ils.unc.edu/grb/ for these examples). The idea was to find ways to make it easier for providers of web content to offer visualizations with their own data.

The GRB is the first example of an open source overview tool that will be available as part of a collection of Information Architecture tools made available by means of an interface server.

Interface Server

There is still much to be done in order to realize a complete agileviews toolkit. The software distribution model is slowly changing from a shrink-wrapped, one time purchase, to online, pay-as-you-go (and only for what you want) system. Many companies, including Microsoft, are eagerly moving towards this new distribution strategy because it promises to alleviate the costs of distribution and upgrades. It also reduces the possibility of illicit copying and cracking. As a Java applet, the Generalized Relation Browser is well-suited to the online distribution model. Future work on the GRB will include the creation of a more robust capability for accessing metadata stored in relational database tables. In addition, other tools that have stemmed from the agileviews framework are in the works. The overall challenge is to develop a framework for how the tools will be accessed and where they will be hosted/delivered. There are several possibilities for how this might be accomplished, any or all of which may be implemented.

All agileviews tools require metadata in order to function. Hence, they are only as good as the metadata, which adds constraints to the design of database schemas. In our experience, it has been difficult to clearly communicate the design constraints of each tool. For example, the GRB needs numeric values (integers, specifically) for its bar displays. Pointing to the wrong type of data, such as a string field, will generate exceptions and result in no information being displayed by the interface. Explaining such nuances, even with detailed instructions, leaves room for misinterpretation and error. Thus, there is a need not only for a comprehensive system for delivering interfaces to users, but also for instantiating them in the first place. Having a wizard-style interface that leads people through the procedure of setting up the GRB, checking for errors and providing feedback, is one way of addressing the problem.

It is probably unavoidable that installing agileviews tools will require a bit more understanding and experience than installing a typical shrink-wrapped software package. We hope to reduce as much of the burden as possible, but recognize that there is a trade-off between providing interfaces and tools that provide agileviews to users, and reducing the burden and complexity on those who will be responsible for the upkeep of those facilities.

Conclusion/Future Work

In sum, the GRB research offers two main contributions to interfaces for web navigation. First, it has proven to be an effective too for helping people explore relationships across collections of data. Its effectiveness is tempered by a few inherent technical limitations, but it works extremely well for small n of attribute sets and attribute values. Second, the generalization experience has demonstrated some possibilities for creating an interface server that allows people to focus on their data and key relationships rather than on the interface implementation and associated details. We see the GRB as a potentially powerful tool for information architects. We will continue to explore similar solutions within the agileviews framework.

Acknowledgments: This work was supported in part by the Asheim Scholarship from the School of Information and Library Science at UNC Chapel Hill. The original development and testing of the Relation Browser was supported by a contract from the Bureau of Labor Statistics. Anita Komlodi conducted the usability tests on the first two Relation Browser prototypes.

References

Brunk, B. (1999). Overview and Preview Tools for Navigating the World-Wide Web, *SILS Technical Report TR-1999-03*. July 31, 1999.

Brusilovsky, P. & Milosavljevic, M. (Eds.) (1998). Introduction to Special Issue on Adaptivity and User Modeling in Hypermedia Systems. *The New Review of Hypermedia and Multimedia*. 4, 1-260.

Bederson, B. & Hollan, J. 1994. Pad++: A zooming graphical interface for exploring alternative interface physics. In *Proceedings. of ACM UIST '94*. 17-26.

Conklin, J. E. (1987). Hypertext: An Introduction and Survey. *IEEE Computer, 20*, 17-41.

Dodge, M. & Kitchin R. (2000). *Mapping Cyberspace*. New York, NY: Routledge.

Geisler, G. (2000). Enriched Links: A Framework For Improving Web Navigation Using Pop-Up Views. *SILS Technical Report TR-2000-02*. February, 2000.

Greene, S., Marchionini, G., Plaisant, C., & Shneiderman, B. (2000). Previews and overviews in digital libraries: Designing surrogates to support visual information seeking. *Journal of the American Society for Information Science*., 51(4), 380-393.

Hert, C. Y Marchionini, G. (1998). Information Seeking Behavior on Statistical Web Sites: Theoretical and Design Implications. *Proceedings of the 61st Annual Meeting of The American Society for Information Science*. (Pittsburgh, PA, Oct. 25-29, 1998). 303-314.

Lamping, J. & Rao, R. (1996). Visualizing large trees using the hyperbolic browser. *Proceedings of ACM CHI '96 Conference Companion* (Vancouver, April 13-18, 1996). New York: ACM Press. 388-9.

InXight Software. (1998). *InXight Software Tree Studio*. Available: http://www.InXight.com. Last accessed Dec. 2000.

Marchionini, G. (2000). Overviews to Previews to Answers: Integrated Interfaces for Federal Statistics. *Final Report to Bureau of Labor Statistics, June 30, 2000*. Available: http://www.ils.unc.edu/~march. Last accessed Dec. 2000.

Marchionini, G., Brunk, B., Komlodi, A., Conrad, F., & Bosley, J. (2000a). Look before you click: A relation browser for Federal statistics websites. *Proceedings of the ASIS 2000 Annual meeting*. (Chicago, IL, November 15-18, 2000).

Marchionini, G., Geisler, G. & Brunk B. (2000b). Agileviews: A Human-Centered Framework for Interfaces to Information Spaces. *Proceedings of the ASIS 2000 Annual Meeting*. (Chicago, IL, November 15-18, 2000).

Marchionini, G., Nolet, V., Williams, H., Ding, W., Beale, J., Rose, A., Gordon, A., Enomoto, E., & Harbinson, L. (1997). Content+Connectivity=Community: Digital resources for a learning community. *Proceedings of ACM DL '97*. (Pittsburgh, PA, July 23-26, 1997), p 212-220.

NASA. (1998). *NASA Home Page Sitemap*. Available: http://www.nasa.gov/siteindex.html. Last accessed, Sept. 1998.

Nation, D. Plaisant, C., Marchionini, G., & Komlodi, A. (1997). Visualizing websites using a hierarchical table of contents browser: WebTOC. In *Proceedings of Designing for the Web: Practices and Reflections* (3rd Conference on Human factors and the Web, Denver, June 12, 1997). http://www.uswest.com/web-conference/proceedings/nation.html

Nielsen, J. (1990). *Hypertext & Hypermedia*. San Diego, CA: Academic Press.

TDED. (1998). *Texas Dept. of Economic Development Sitemap*. Available: http://www.tded.state.tx.us/sitemap.htm. Last accessed, Sept. 1998.

Robertson, G., Czerwinski, M., Larson, K., Robbins, D., Thiel, D., Y Van Dantzich, M. (1998). Data Mountain: Using Spatial Memory for Document Management. *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology* (San Francisco, Nov. 1-4, 1998). 153-162.