mSpace: interaction design for user-determined, adaptable domain exploration in hypermedia

m.c. Schraefel (1, 2)
(1) Dept. of Electronics and Computer Science, University of Southampton, Southampton, UK,
mc at ecs.soton.ac.uk

Maria Karam, Shengdong Zhao (2)
(2) Dept of Computer Science, University of Toronto, Toronto, Canada
{mc, mkaram, sszhao}@dgp.toronto.edu

Abstract

Adaptive Hypermedia systems have sought to support users by anticipating the usersí information requirements for a particular context, and rendering the appropriate version of the content and hypermedia links. Adaptable Hypermedia, on the other hand, takes the approach that there are times when adaptive approaches may not be feasible or available and that it would still be appropriate to facilitate user-determined access to information. For instance, users may come to a hypermedia and not have a well-defined goal in mind. Similarly their goals may change throughout exploration, or their expertise change from one section to another. To support these shifting conditions, we may need affordances on the content that a solely adaptive approach cannot best support. In this paper we present an interaction design to support user-determined adaptable content and describe three techniques which support the interaction: preview cues, dimensional sorting and spatial context. We call the combined approach mSpace. We present the preliminary task analysis that lead us to our interaction design, we describe the three techniques, overview the architecture for our prototype and consider next steps for generalizing deployment.

Categories and Subject Descriptors

H5.4 Hypertext/Hypermedia-Architectures, Navigation, User issues. H5.2 User Interfaces-Prototyping. General Terms Design, Experimentation, Human Factors

Keywords

Hypermedia, Task analysis, adaptable hypermedia, information visualization

1. Introduction

One of the promises of the Web, and more recently the Semantic Web is greater access to information. Make it digital: make it accessible. Representing digital information, however, is a huge challenge: make it digital, it often becomes invisible. The problem has been approached from multiple angles. Adaptive Hypermedia research in particular focuses on getting the right information about a given topic to the right user • whether that user is a beginner in a domain or an expert[1]. Information Visualization research focuses on the challenges of presenting rich information sources on (mainly) visual displays, dealing with challenges like finding the balance between how to balance contextual information with information currently in focus[7]. Both Human Computer Interaction (HCI) work and Hypermedia research has considered problems of navigating through hyperspaces.

Visualizing, navigating, and determining user level might each be seen as part of the information problem after the information source has been discovered. In this paper, we consider the part of the problem just before this: the issue of accessing the information in the first place. How can we reduce false steps in locating appropriate domain information? And then, in the sense of hypermedia affordances, how can we provide the support for the user to engage the information, to support their making their own connections about the materials discovered?

In this paper we propose an approach for domain interaction to support both discovery and connection building. We do this with three core interaction functions: preview cues, dimensional sorting and spatial context. We call this interaction approach mSpace (for representing multidimensional spaces). In the following sections we present the motivation for mSpace. We describe the task analysis evaluation we used to determine the interaction requirements, and detail the resulting prototype we have developed stemming from that analysis. We briefly present the architecture behind the current prototype, but emphasize that our interest is on the interaction approach: to design hypermedia-informed affordances for an information delivery system; we are architecture/backend agnostic. We discuss the related work as it applies throughout each section of the paper.

The techniques we describe afford the community new, additional methods by which better access to information can be achieved fro more users.

1.1 Motivation: Search, Browse, Exclude

Google has become the default method for accessing information on the Web. One enters their keywords for a search, a list of resulting links is returned, one browses through the results to find pages which best satisfy the query. For most cases, this is a highly successful method for retrieving information. There are some kinds of queries, however, that Google cannot support, such as when the users are domain naive. That is, these users do not have either the expertise to frame a query to begin with, or to interpret the results were they to be presented with them. For instance, people who know little about classical music, except that they know what they like when they hear it, cannot use Google (or any keyword search tool) because they do not know either the names of the pieces or the terms by which they are described. Similarly, if they do a generic search for Classical Music, they would find little value in browsing through sites with comprehensive listings of all recorded classical works: descriptions such as sonata, serenade, romantic or rondo are equally meaningless if all the user knows is what they like when they hear it. In effect, if they cannot name it or spell it or specify it, they cannot access it.

To begin to address the information invisibility problem, we proposed the following: first to represent information in terms of a spatially represented domain, an organized representation of an area where elements represented within the domain have defined associations to each other; and second, through the interaction, to leverage what a user already knows about that domain to support access to it. In the case of classical music for instance, users are presented with a structured representation of the classical music domain with categories such as Period, Genre, Composer, and Instrument. On their own, these categories may be meaningless if users only know what they like when they hear it. Therefore, we associate music with each attribute of the domain so that users can determine if they are interested in exploring this dimension of the domain based on what they can assess: whether or not they like the music associated with that attribute. We call this technique Preview Cues [11].

Similarly, if users are more familiar with instrument than composer they can rearrange the domain so that instrument is the primary dimension, with other dimensions projected as dependent levels of a hierarchy descending from instrument. We call this technique dimensional sorting. These techniques allow people to leverage what they know about a domain to enable discovery from their frame of reference rather than a particular information sourceís. In this approach, the domain is presented in a structured way, and the potentially unfamiliar structure is made explorable by preview cues and dimensional sorting that enable the user to make a judgement about how they wish to proceed through the space. We hypothesize that through the combination of these techniques • structure, cues and sorting - users will begin to learn the domain • gain domain expertise • simply by interacting with this domain interface. We further hypothesize that the ability to manipulate the domain from multiple user-determined perspectives supports discovery of knowledge within the domain that may not otherwise be readily apparent. In the classical music example, for instance, people simply using the interface would be able to learn implicitly that, while Bachís keyboard works are often recorded on the piano, the piano itself was invented after Bachís lifetime. One may then want to know what his music would sound like performed on the instruments of his time. With the interaction proposed here, the user could make these comparisons easily by moving from Works to Recordings. Thus, an initial discovery may lead to a new query and there to a new discovery, all of which can be satisfied from the usersí frame of reference.

We carried out an initial study to look at how combining a spatial domain representation with preview cues would improve access to classical music [11]. Based on the success of this work we considered a a more abstract problem space, JavaScript documentation for various levels of expertise, which lead us to the development of dimensional sorting. The combination of spatial context, preview cues and dimensional sorting is proving to be a gestalt technique for improved information access in adaptable hypermedia. We ground the presentation of the techniques first in a discussion of the task analysis we performed in the problem domain.

2. Task Analysis

We chose the JavaScript domain for two reasons: the classical music space trials proved to be expertise-neutral in terms of performance and enjoyment. Both experts and novices found the tool effective and satisfactory. We werenít sure if these results would carry through to a domain context where the goal is more complex information support, in other words, where more than simple selection based on preference is involved. In the case of JavaScript documentation, users from various expertise backgrounds would have distinct problems to solve, based on distinct levels of previous knowledge. We wanted to see if we could develop an effective adaptable interaction space that would support this range of expertise and complexity. Before we designed our interface, we needed to understand how JavaScript practitioners approached problem solving with current resources. It is absolutely critical to perform such preliminary evaluation of the problem domain, without which any model we proposed would be shooting in the dark. To this end, we performed a task analysis. Task analysis is a method in HCI [8, 14] which can be used to break down a task into goals and subgoals, show their order of execution, and make explicit any dependencies among goals. The resulting task model allows us to see where inefficiencies in the interaction exist. From this analysis, we can develop approaches to address the problem.

2.1 Task Domain

In our case we were interested in assessing how users used online resources to assist solving JavaScript problems. JavaScript is one of the most common scripting languages used for

programming interactive effects into web pages. Based on our own experience with Web site design and Web application development, we predicted three user profiles: (1) Web site designers who use JavaScript as part of prewritten effects available in packages like GoLive and Dreamweaver. (2) casual JavaScript implementers who were either programmers, but without much JavaScript experience, or web site developers who were capable of editing prewritten JavaScript code, and (3) experts who used JavaScript coding professionally. After interviewing twelve participants who identified themselves as people who worked with JavaScript code, we found that these profiles were appropriate stereotypes, and could be further coded as Novice, Intermediate and Expert users. However, what was of particular interest to us is that each group required quite distinct JavaScript resources from the other. This diversity of user types gave us a different problem approach for user access to domain information than the user modeling case in Adaptive Hypertext (AH) research. In AH work, particularly educational AH [3], we are most interested in creating user models to adapt the same set of information in different ways as appropriate for different levels of expertise. In JavaScript, however, different user levels require different kinds of JavaScript information. Our challenge was to see if there would be value in (a) modelling a single site to support all these expertise resource needs and (b) interlinking all the information components in a manner to support a rich context for any user's access of the site at any entry point.

2.2 Task Evaluation

In order to understand both what kind of information requirements our three users profiles needed for the kinds of JavaScript problems they had to solve, we used two types of Task Analysis approaches: participant interviews about their work with JavaScript and an experimental evaluation of task performance.

2.2.1 Experiment

The study involved 12 JavaScript users, 4 novices, 4 intermediate users and 4 experts. Each participant was given 30 minutes in which to complete a task that required them to add a JavaScript feature to a sample web page. These tasks were based on a pilot study of 5 participants to determine what an appropriate task for each user level would be. Novice users were asked to add a script to a page that would make a banner appear with an imaginary company's name. Intermediates were asked to create a popup window that sizes itself to 60% of the available screen size of the user's system. The expert users were asked to create a trim function that removed all the leading and ending spaces from a string and displayed the new string. In each case, participants were shown a running version of a page using the code for their task, so that they could clearly see what they were being asked to do. They were also given a web page template into which they could insert their code for testing. The same computer was used for each session, and the same tools were provided: a text editor for code writing and editing and a web browser for web searching and code testing. A pre-task questionnaire helped determine the users' expertise level and a post-observation questionnaire discussed problems they encountered during their tasks. Participants frequently made suggestions for improving resources to solve those problems. Each participant was also encouraged to "think aloud" to describe what they were doing as they worked on their tasks. Each session was also taped.

2.2.2 Observations

2.2.2.1 Resource Use and Expertise Approach

We observed a consistent set of steps across user groups to solve the task to Get a Script Running. There were two large subgoals: (1) find the appropriate resources; (2) test the result.

There were multiple subgoals stemming from these two core goals. The model is presented in Figure 1, below.

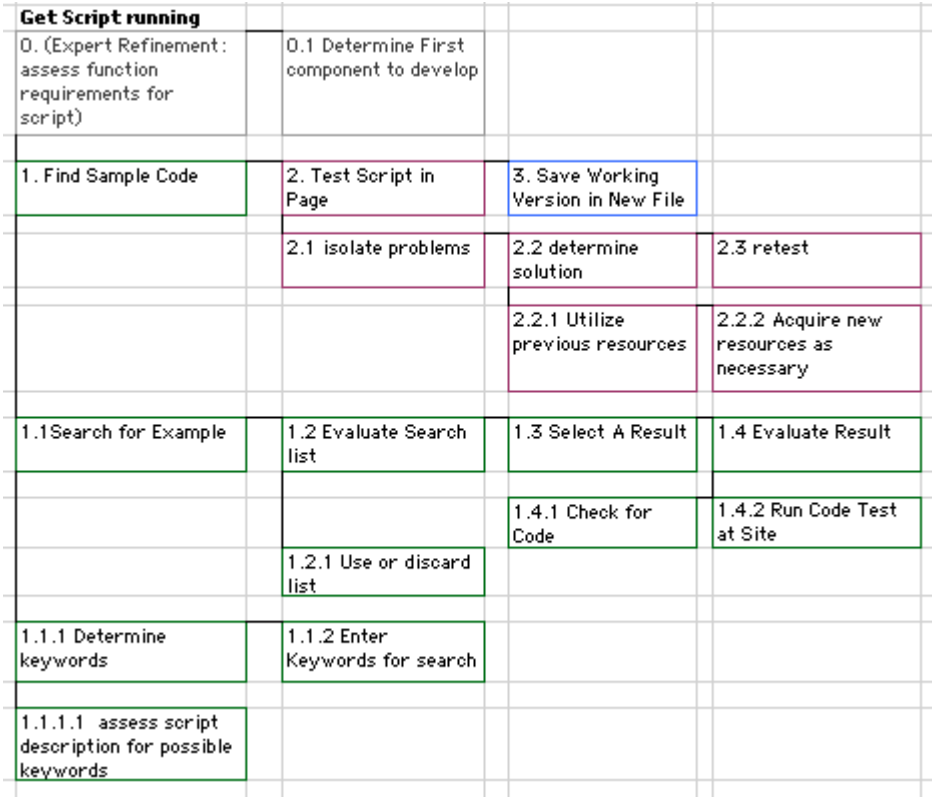

Fig. 1. Hierarchical Task Analysis of "Get Script Running"

The model is based on Hierarchical Task Analysis[4]. This approach lets us focus on the task and its subgoals without making assumptions about user knowledge structures. As can be seen from the diagram, most of the subgoals relate to the task of resource discovery. Subtask 1 "Find Sample Code" has ten subgoals compared to the five steps of Subtask 2 "Test Code'". The model was the same for each user level, with the exception of one preliminary subgoal introduced by the expert users (labeled 0 and 0.1 above) to pre-assess the problem for what function calls it would likely require.

In subgoal 1, users either went directly to a source of information that they though had the answer to their questions, such as a JavaScript or design site, or did a more general key word search to begin. Key word searches took place either at the Web level, or within a JavaScript site. The second step, once the resource was located was to start coding, either by hand in the text editor, or by copying code directly into the web page template. If the code failed to produce correct results, users returned to step 1, either refining their search to find a better resource, or conducting a different search to address some problem they encountered while implementing their code.

Novices and Intermediate users searched for working examples of JavaScript that they could view and then copy into the Web page. Advanced users already had significant personal JavaScript knowledge, and found it more efficient to craft their own code. Generally, their need to consult resources was at the syntax level, so language references were important. They expressed wanting to understand how the code worked at that language level. If they encountered a bug in their code they could not solve from this approach, then they said they would look to Q&A sites to see if anyone had encountered the same error and what the solution was. None of the experts, however, found this necessary for the given task.

2.2.2.2 Number of Web Pages and User Level

We captured both sites visited and frequency of sites used by recording the browser history for each participant during the task.

Table 1. Pages per User Level During 30min. Task. U=Unique Pages; HI = Highest number of return visits; U to HI = ratio. Only results of .5 or higher are rounded up.

	Av Hits	Av U	Av HI	U to HI
Novice	35	24	4	6
Intermediate	26	18	6	3
Expert	26	11	5	2
Overall	29	18	6	4

Novices returned to the Web about 25% more than frequently than Intermediates or Experts, who used the web about equally. Experts, however, used about a third fewer unique sites. Also, as expertise increases, so does the ratio of reuse to discovered resources. Novices, on average, reused a sixth of their repeated resources, intermediates, a third and experts, a half.

2.2.2.3 Task Success

Those users who completed the task required about two-thirds the time allotted. Only 2 users did not finish the task: these were both novice users whose only previous experience of JavaScript had been using prewritten functions in a Web design program. Despite this, they expressed confidence at being able to complete the task, having viewed a considerable amount of HTML source with JavaScript embedded. Indeed, they successfully found code that provided the correct function, but the source did not provide a sample page, demonstrating the function call in the body of the web page.

2.2.2.4 Particular Domain Access Problems

Below, we summarize the observations we made during the task analysis. The following is based on observing participants while they worked, the comments they made while "thinking aloud" during the task, and the post task interviews.

1. Users experienced difficulty in articulating what they wanted for keyword searches within the Web resources.

Locating information about JavaScript can be an exercise in working around potentially ambiguous terms and concepts. Unless the user is experienced with the terminology or gets lucky, locating the exact concept is difficult. For instance, coded example titles are not necessarily associated with any JavaScript concept: searching Window Event does not necessarily lead to Resizing Windows as an associated topic. In other words, Google did not work effectively for retrieving ìan answerî

2. Users were not aware of useful resources that could help them solve their problems.

A common problem for novices and intermediates in particular was lack of awareness or discovery of available online resources. For instance, users across expertise levels stated that

they thought a reference manual for JavaScript would have helped them with their task. Interestingly, they did not search for manuals during their task, and several exist, including Netscape's very popular JavaScript resource.

3. Users from different experience levels sought different kinds of resources for solving similar problems.

Although the tasks were very similar for the different expertise levels, the approaches that the users took to solve their problems varied with the amount of knowledge that each user had. Novices were aware that code examples existed online, so they sought these out for direct copy and paste. If however, the examples did not provide instructions about the function call, the novices were unable to succeed. Intermediate users were happy to discover working code samples, but also connected what they were attempting with previous knowledge about programming, and evinced little hesitation in attempting to alter the function given or its call. If the function did not work on first pasted into the Web template, they were able to hack the code to get it to work. If they had problems with the code, they would look for a resource to explain part of the code. This caused some difficulties narrowing down what part of the code was the problem and which reference was needed. Experts stayed consistently at the level of syntax in their searches, generally looking first to see if the function they needed was already part of the JavaScript set of objects. One expert only had some search time difficulty determining which object string trimming was part of (it's "string"). From there, they could hand build the code for the page.

4. Users were frustrated by the poor quality resources and code examples that they had to work with.

All of the JavaScript users began their task by searching for examples of code or functions that they could use as a reference. All of them also complained that much of the code that is available is poor quality and unreliable. Since grabbing code samples is one of the most common means of building JavaScript code for our novice and intermediate participants, those without backgrounds in programming usually have no way to check the accuracy of the code. Experts seemed to avoid most of this problem by searching for appropriate functions from which to build there own code. More so than intermediate users, they first analyzed the problem, and broke the problem down into likely component requirements. They then searched for those functions/objects first if no sample code was immediately available from a keyword search.

2.2.3 Analysis

With both the task model and the Pages per Use data, we are able to consider any correlation that may exist between number of iterations through any part of the model and user expertise. We can use these correlations as guidelines for where functional improvements in the interaction design need to be considered. The Pages per Use table (Table 1) above suggests, perhaps not surprisingly, that Experts are best able to discover and narrow in on a particular set of resources as they work, and reuse those resources (subgoal 2.2.1), flipping back and forth between a narrower set of mainly cached pages, than the other groups. Novices were least well able to do this. They cycled through the steps of subgoal 1 far more frequently before moving to subgoal 2, testing the code. Despite efficiency of resource use by experts, there was not much difference between the number of times all user levels reloaded pages. An average of 29 discrete reloads during a 30 minute task is considerably high. Each reload of a page replaces the previous window, the previous context, requiring the user to bear increased cognitive load to maintain that context. For novice users, based on the number of unique pages overall they touched, it is apparent that they had difficulty even establishing a context.

Not only do multiple page loads compromise a sense of persistent context for support in a task, they take time away from other subgoals. Even when pages only need be glanced from a list of returned search results to determine whether or not to pursue them, there is time involved in selecting a link to load, time for load, time for page scanning, dismissing and returning to the search. Overall, then, the main areas for task support improvement are in resource persistence to decrease searching through a stack for already discovered, preferred resources. Better presentation of the domain itself is also required to help novice and intermediate users in particular overcome barriers to access that key word matching imposes, causing them to look more frequently for new resources, rather than reuse existing ones. The following section presents an interface prototype and its design and function rationalization to address these problems.

3. Interaction

From the analysis above, we see the global requirement is, as suspected, to improve access to domain information, but also to provide a way to support the particular usersí perspectives, or frame of reference, on the data. Our interface proposal, therefore, is first and foremost to represent the JavaScript domain in a more accessible way for the range of users. Since much of the time in the above task model is spent simply attempting to locate resources, providing a single resource as an information broker and domain context should improve task performance for help with this part of the interaction. However, within this representation we must also support the userís context.

3.1 Improved Context = Spatial Context

Research in information visualization, referenced below, suggests that spatial, hierarchical visualizations of information are more effective than list views • the default views of the Web for information discovery. We leverage this research and expand it to include hyperlinked supplemental information. In this way, we present more than a spatialized view of information contexts, but also in-context information about path elements to assist the user in making navigational decisions. This gives us a tool to investigate the combined approaches of spatial views with artifacts similar to link annotations.

3.1.1 Context for Context: Link Lists

Most Web searches return lists of linear results, and usually limit the maximum number of results displayed to some interval, as per Figure 2, below.

Expanding/contracting hierarchical list views, common in most desktop file system navigators (Fig 3), have been shown to be better for most kinds of information retrieval than the list [3].

As with [3], we have also found strong results with a multipane type display which expands the contents not into the same list, but into a new adjacent frame, where each frame can be scrolled independently [10]. A version of this kind of viewer was made popular in the NeXT OS, and is used more recently in the Mac OS X environment (Figure 4).

Expanding and collapsing lists have the advantage over lists of maintaining a controllable, persistent context for the user. This persistent context is also interactive: a much or how little associated information in a hierarchy appears in view. Chimera et al [2] found that this control reduced scrolling through data and improved discovery efficiency as users controlled the area of the data on which they wished to focus. We also found [11] that users reported having a

better sense of context with multipane views than with single list views of part of a hierarchy, which placed path information above the list. This path above list approach is common in Web category sites (Figure 5).

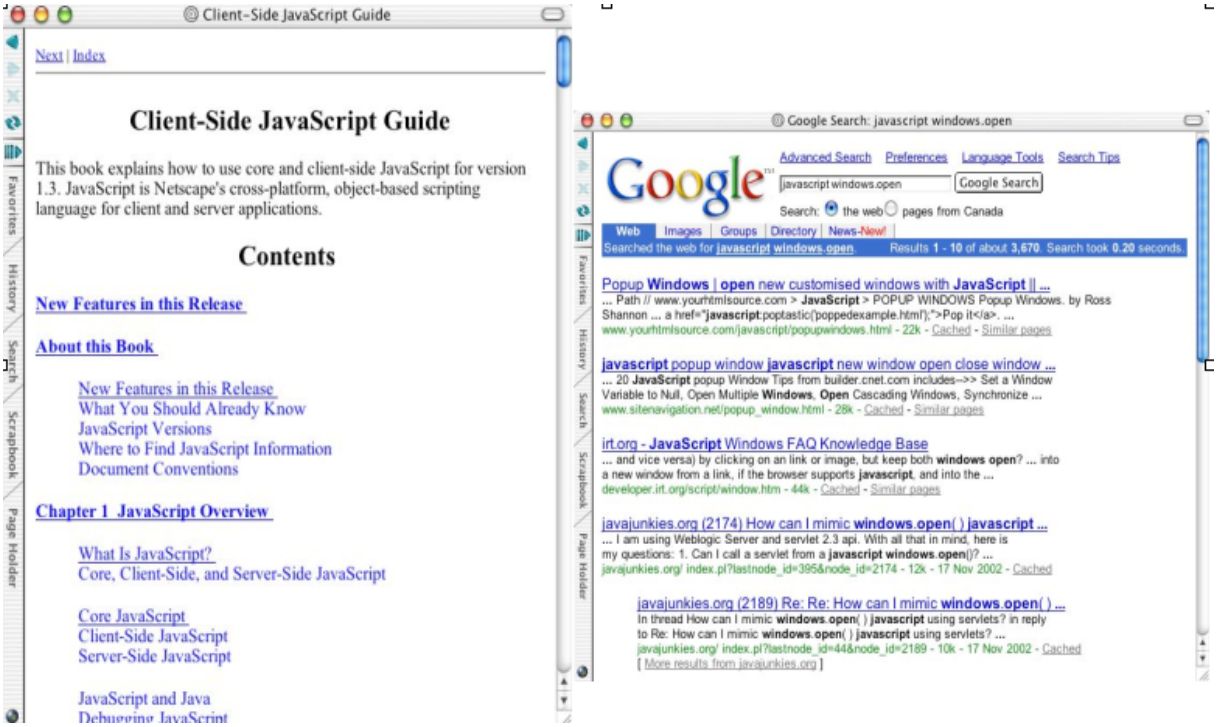

Fig. 2. Two common Web page examples of linear lists for accessing content: the left window is linked TOC for an online manual; the right is a search result window.

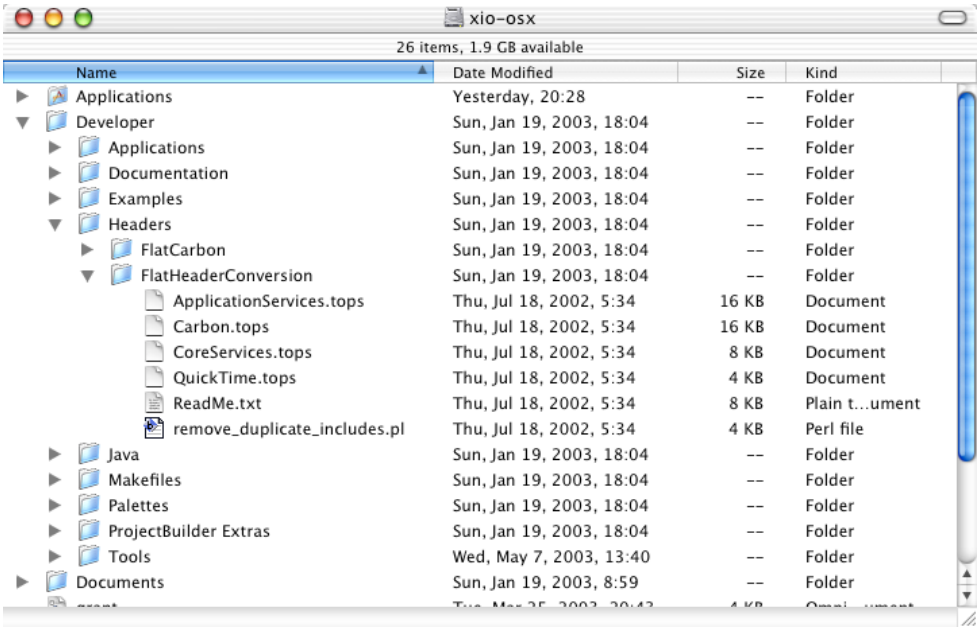

Fig. 3. Example of an Expanding/Collapsing list shown as indented) or collapsed (shown in line).

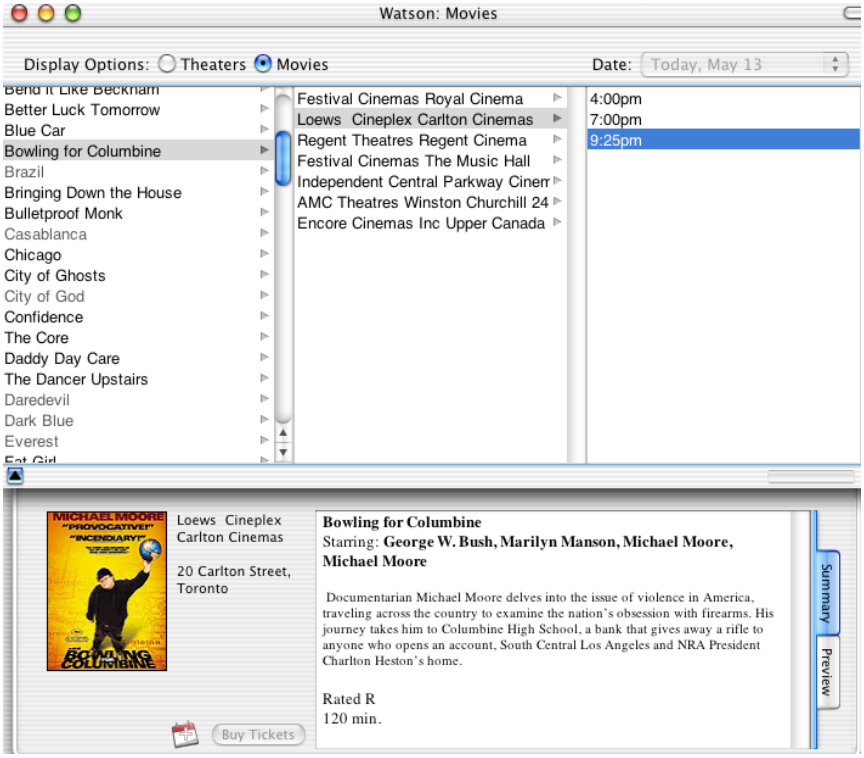

Fig. 4. Example of a multipane file browser in which items expand into the adjacent pane (Watson, Carelie Software). A type of detail rather than preview cue is also available in the lower pane description of the film. In this case, the detail reflects a specific instance, (the theater location and the film description). The detail here is very specific: films showing in Toronto, without associating information that describes kinds of films,etc. That said, the power of bringing this information together in one location, with ease of exploring options like locations and times, is undeniable.

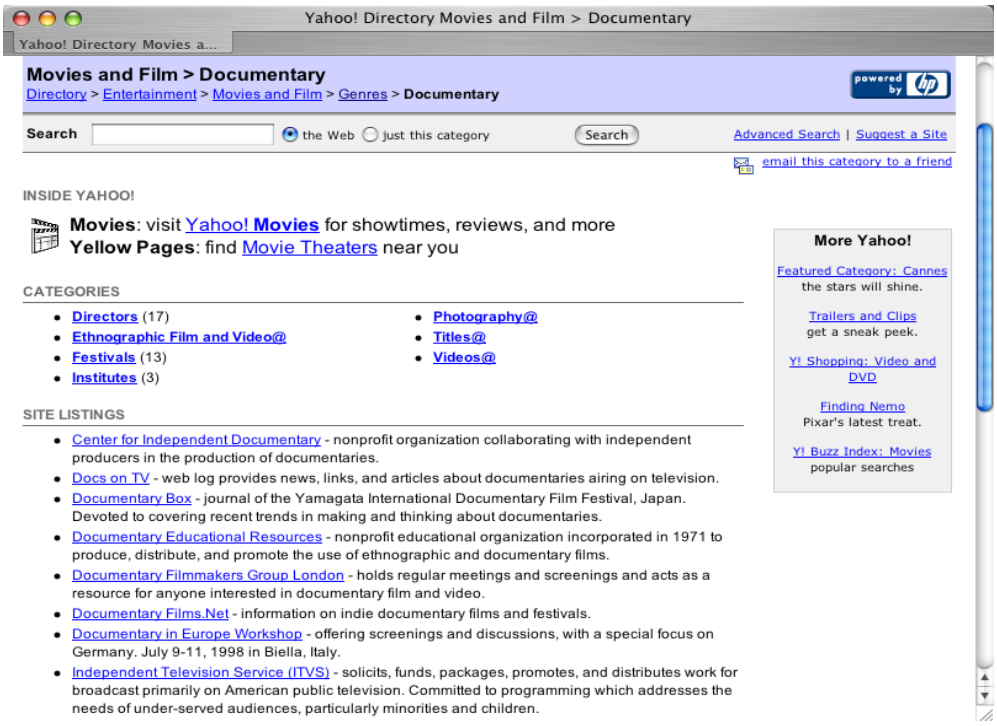

Fig. 5. Sample Web page in which path to current node in hierarchy is given via a category list (page top). Links for the current node are viewable in the vertical list beneath.

We also found that, especially if users have discovered an item of interest within a hierarchy, they prefer to see the context for each part of the path to their current location. A multipane viewer such as Fig. 4 provides this context by highlighting the previous node, situated among the visible list of nodes/leaves in that pane/at that level of the hierarchy. While we have not tested this explicitly, observations suggest that this persistent, node-level context assists in building a mental map of the domain. Such a map assists in information discovery and rediscovery.

We have extended the multipane browser to work as a hypermedia multipaned browser. For these reasons of demonstrated value for context support and efficient information discovery, we adopted a multipane hierarchical topic browser approach for our JavaScript domain interface prototype (Figure 6, below).

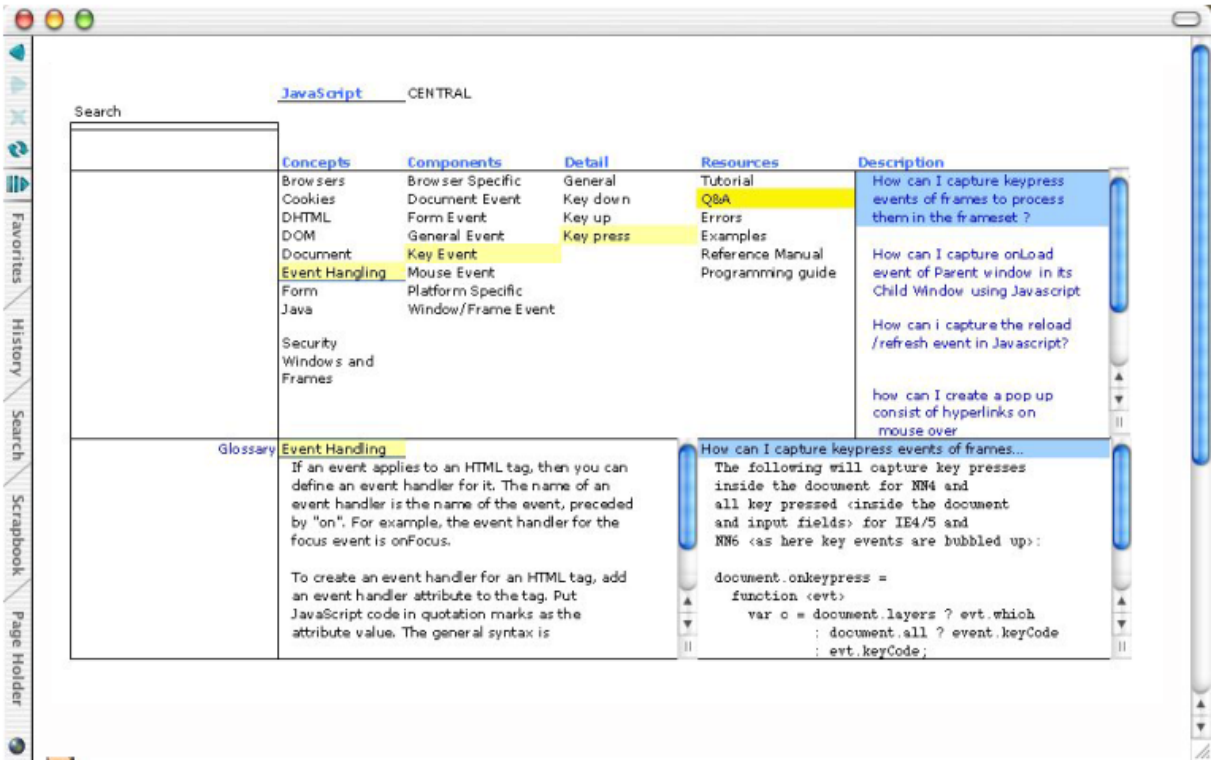

Fig. 6. JavaScript Central mSpace prototype

Context is maintained several ways: in the main area of the browser, topics in JavaScript are revealed in a multipane hierarchal view. As the user clicks a term in one pane, two events occur: (1) the next layer of hierarchy opens in the adjacent pane and (2) the glossary pane beneath the browser is populated with the entry for the selected term. By option clicking on the term in its pane, the user can "fix" that term in the glossary. A fixed term is underlined in the pane (as is Event Handler in Figure 6). If the term is not fixed, the glossary will reflect the last level of the hierarchy selected which has an available entry. In the example in figure seven, this would be the term "key press." In the final pane of the browser, the user is presented with a list of discrete entries for the specific topic. Clicking on one of the entries opens that item in the detail pane beneath that list.

Each of the panes such as the glossary entry, detail view, and item list can be opened into their own windows. Clicking on the term "Glossary" also opens a complete glossary-listing window that the user can scroll through, using an expanding/collapsing list for topics/subtopics. In other words, users have control over how much information they want displayed at any time.

The Glossary and main browser are also interlinked: clicking on an entry in the glossary window highlights its path through the multipane browser. This allows users to move between glossary and context view to explore relations among data elements. Similarly, the search box in the upper left of the window produces a list of results, organized by topic, and displayed in the blank paned beneath the search box. Clicking on a result in that list highlights its path through the multipane space, leaving the final Description pane blank.

Finally, for context support, we have incorporated work from a previous project, Hunter Gatherer [12]. With Hunter Gatherer as part of the architecture, users can select any component within any area of the browser or any of its windows, and have those selections collected into a new window. These collected components can be saved and shared with others. They also act as a reference resource so that, rather than flipping back and forth through collected resources while working on a problem -- as we saw our experts do • users can have in one place the information they found useful. Users can create multiple collections of resource components for future reference as well. Each collected component in a collection window maintains a link back to its source information block. Users can return to this larger source any time they wish to peruse the larger context of a reference. Each source document (what we call and information block) has markers associated with it, identifying other areas in the domain to which it can be associated. Clicking on any of these codes again highlights that path through the multipane viewer so that users can quickly perceive associated contexts and explore these contexts should they wish.

In our prototype, a user does not click out of the main window. The data context remains in view. When a new window opens, it does not completely occlude any other, and the user can always resize or dismiss that window. Detailed views are available quickly, and in context, supporting faster evaluation of any list of entries returned in either the search pane or the description pane. Information found to be of value can also be readily collected and maintained in its own window for persistent availability.

3.2 Improved Access

Preserving context, we know, is beneficial for discovery and navigation through information sources. In the above section, we describe several techniques that we have employed to give users both control over how they see the information in the domain, as well as how they are able to access the larger context of any component from multiple information locations in the interface. Context preservation, however, is only valuable once a user gets to the domain and begins to investigate a path. How does a user • a domain-naÔve user in particular • determine which path to explore, especially if the domain terminology is unfamiliar? Part of the task breakdown for novice users in our study was lack of term knowledge: they spend considerable time hacking around web pages looking for jargon that matched what they saw before them on the screen. Few of their discovered resources assisted them in narrowing their focus, since few of these examples showed any relation to any other example/context. Even presented with our multipane view, a domain naÔve user may spend time doing a brute force search through nodes of the tree [2]. Arbitrarily going down paths is hit and miss, not informed exploration, particularly if the domain is large. To reduce this kind of domain hacking, we developed Preview Cues and Dimensional Sorting.

3.2.1 Preview Cues for Pre-Path Evaluation

Preview Cues reveal information to describe a range of values referenced by its label. The cues are triggered by a user holding down a command key while brushing their cursor over a term in one of the browser panes. In this way, users get a sense of a range of a domain before

they commit to exploring that part of a domain. Unlike tool tips or Fluid Links[18], domain preview cues do not have a one to one correlation between themselves and their trigger. A preview cue for Event Handler is not a definition of a term (that is provided in the glossary pane). It is a representation of a range of values, from which a sense of that part of the domain can be derived. Showing domain range through preview cues is not unlike multi-pointing links: rather than have a link go to only one document at its end, multiple associated documents can be pointed to. Such lists, however, represent all the possible links the system has available. They are also possible paths from which users can head out further; preview cues point in: they are representative samples of all available information in that part of the domain. For instance, a user control-brushing over the label ìEvent Handlerî causes a new small window to open to the right and top of that term. The window contains (in the current prototype) three entries representing event handlers: a representative question asked about events, a code description of a specific kind of event and an example of working event code which the user can run in that preview window. No one had to handwrite these domain previews; they are taken directly from information available to the interface from the local domain database. We will come back to this briefly in the Architecture section below. By clicking on any of the elements in the preview window, the user will see where in the path the preview is from. Along with the glossary entry below the multipane browser, the preview cue quickly provides information to users for immediate evaluation of that node in the hierarchy. Users can then decide whether or not they wish to explore further along that path. If the user brushes on a deeper part of that same path, such as "Key Press" all the preview cues would reflect the particular part of the domain relating to the Key Press event. The location of the brushed element in the path acts as the lower bound on the region for preview cue building.

In this respect Preview Cues are closer to Side Views [16] than to Query Previews[6], but both are worth mentioning. In Side Views, users of image editing software can preview simultaneously the effects of a filter: multiple versions of the image are given which show what it would it would look like with different settings of the filter applied. Seeing many versions at once saves the user from going back and forth between the image, trying one filter setting, discarding it and trying another and so on. With multiple views, the user has more data available in context to determine whether or not the filter is appropriate and with what settings. Preview Cues act likewise to give a sense of a range of values for determining whether a path is the right one to explore. Query Previews, on the other hand, provide users with preset sliders and buttons on data which the users operate to see a all the data that matches the input values, should a result exist. Preview cues are situated not to afford specific results but to act as domain indicators.

In our study of domain preview cues [10] we found that while 95% of the 82 participants in our gender/age balanced study found preview cues useful for refining path selection, and that they wanted them available. 50% of users were satisfied with preview cues being turned on with any brush over a term. 45% of participants, however, wanted to be able to turn them on or off as they chose. This is consistent with Zellweger who also found in the display of annotations on mousing over links in documents that user control annotations was critical [18]. That is why in this prototype we use a control key to modify the brush over.

Preview cues give users examples of domain content that they can quickly evaluate whether or not that area is of interest for perusal. Because these cues are assembled from pre-existing components in the domain, no special content needs to be written to describe the range of possible values in the domain. At this time, we are working on developing specific heuristics for determining best or most representative samples for a domain elementís preview cue. In the interim, what we have found is that any preview cue available on request is significantly better than no preview cue.

3.2.2 Dimensional Sorting for Orientation

While preview cues help users make decisions about exploring part of a domain, dimensional sorting lets the user organize the domain in a manner that suits their current interests/knowledge. Providing multiple perspectives on an information space has long been valued by hypertext researchers not only for access for but the value of learning more about a domain [5, 15]of building knowledge by seeing associations that would not otherwise be apparent. It also lets the explorer move into the domain from a locus of familiarity. For the domain-naive user, dimensional sorting may provide an approach into a domain, without which they would not be able to enter. In the JavaScript example, an expert user may find it effective to explore the domain in the way it is organized in Figure 6, privileging Language. A domain-naive or novice user may want to see the domain organized to privilege Examples, in order to see simply what the language can do. Our interface supports such rearrangement.

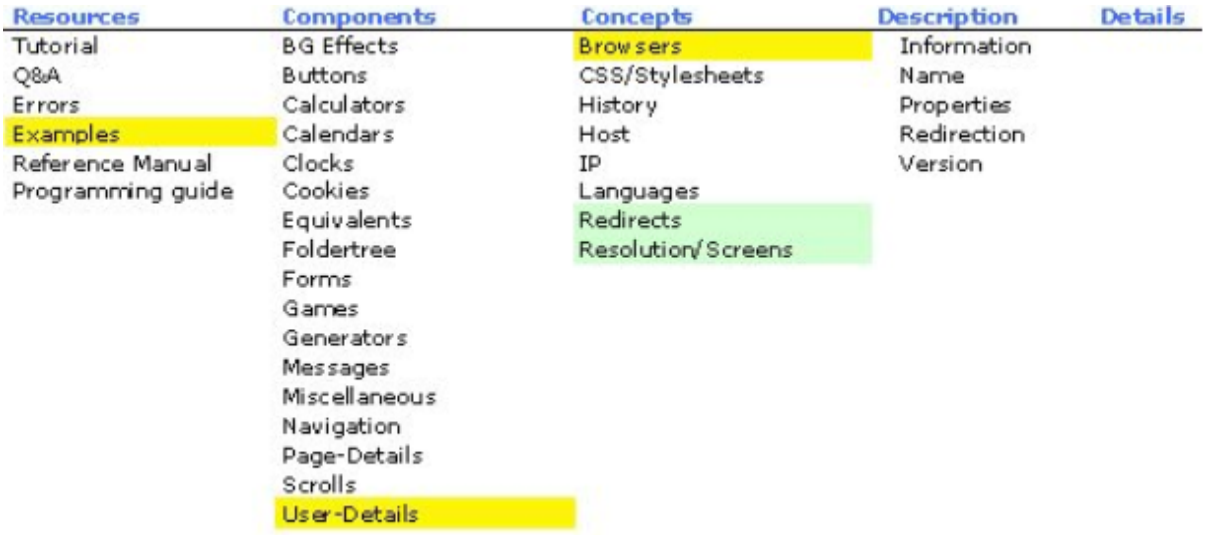

Fig. 7. The JavaScript domain hierarchy reorganized.

Hierarchical rearrangement is not to be confused with sorting order within a fixed hierarchy. Rearranging a hierarchy causes elements to be regrouped. Indeed, dimensional sorting is named for seeing these domain spaces as n-dimensional spaces. In order to visualize these n-dimensions, we flatten the space as a projection along a dimension. This allows us to render the space as a hierarchy. Flattened in this way, we can apply hierarchical visualization techniques to render the hierarchy in multiple ways. The multipane view we have chosen here is one among many possible visualizations that could extend to two or three dimensions. The mSpace model does not insist on any particular visualization; the only constraint we impose on the visualization is that it supports persistent context. In the case of dimensional sorting, the multipane view affords a clear persistent context that makes explicit the effects of reorganizing the categories of the domain. In the above example, for instance, we have pulled Resources from the fourth position to the first and moved the Concepts category from the first to the third position in the order. This rearrangement supports users who primarily want to consider code examples in the domain. Users could get at all the examples in the domain in the previous hierarchical organization, but to do so, they would need to move through almost each element of the hierarchy iteratively to gather all the examples they would want. By supporting dimensional sorting to improve access from multiple perspectives, we are not explicitly modeling users for a particular version of a domain, but are letting users determine the domain version for their needs/tasks.

3.2.3 Adaptable vs. Adaptive Hypermedia

In terms of their hypermedia context, both Preview Cues and Dimensional sorting can be considered as types of adaptable [9] rather than adaptive hypermedia techniques. They emphasize user-directed support to adapt domain representation rather than system-determined adaptive representations based on user-modeled domain interactions. The approach provides tools for users to determine directly the perspective from which they wish both to explore as well as to orient a domain. Preliminary assessment suggests that it is well worth considering integrating such adaptable direct-manipulation techniques with adaptive approaches.

3.3 Preliminary Assessment

3.3.1 Design Reviews and Predictions

The above prototype is the result of cognitive walkthroughs and design reviews. The walkthrough let us test the design flow and prototype responses to user actions via evaluations of paper-based mockups before coding the above prototype. We have only just started testing the prototype with real users at multiple expertise levels. In casual presentation with domain users, feedback has been positive, with participants keen to know when the site will "go live." Once we have a larger data set behind the prototype, we will be able to do the next round of user evaluations to see first, if our new model is more efficient and second, if we have broken the reliance on having to formulate a keyword search and improved efficiency/effectiveness in accessing this domain space.

4. A Word on Architecture

In order to support the above interaction prototype, we developed a functional back end that let us demonstrate the interaction for evaluation. We do not postulate this as a ìsolutionî for supporting the described interface in general practice.

The architecture, however, has let us process and repurpose a considerable amount of Web-based data from multiple sources. In future, this processing is an ideal job for the Semantic Web. Indeed, we have already taken the lessons learned here and applied them to a Semantic Web ontology/backend [13]. That said, the architecture is functional for repurposing existing web-based information sources with minimal manual involvement.

The main purpose of the architecture is to gather JavaScript information from a variety of sources which can be represented and interacted with in one place. This domain-based approach allows users to maintain a persistent context, rather than the user having to jump from site to site to find what they may be looking for. The associated domain approach supports evaluating any particular information component within a variety of contexts.

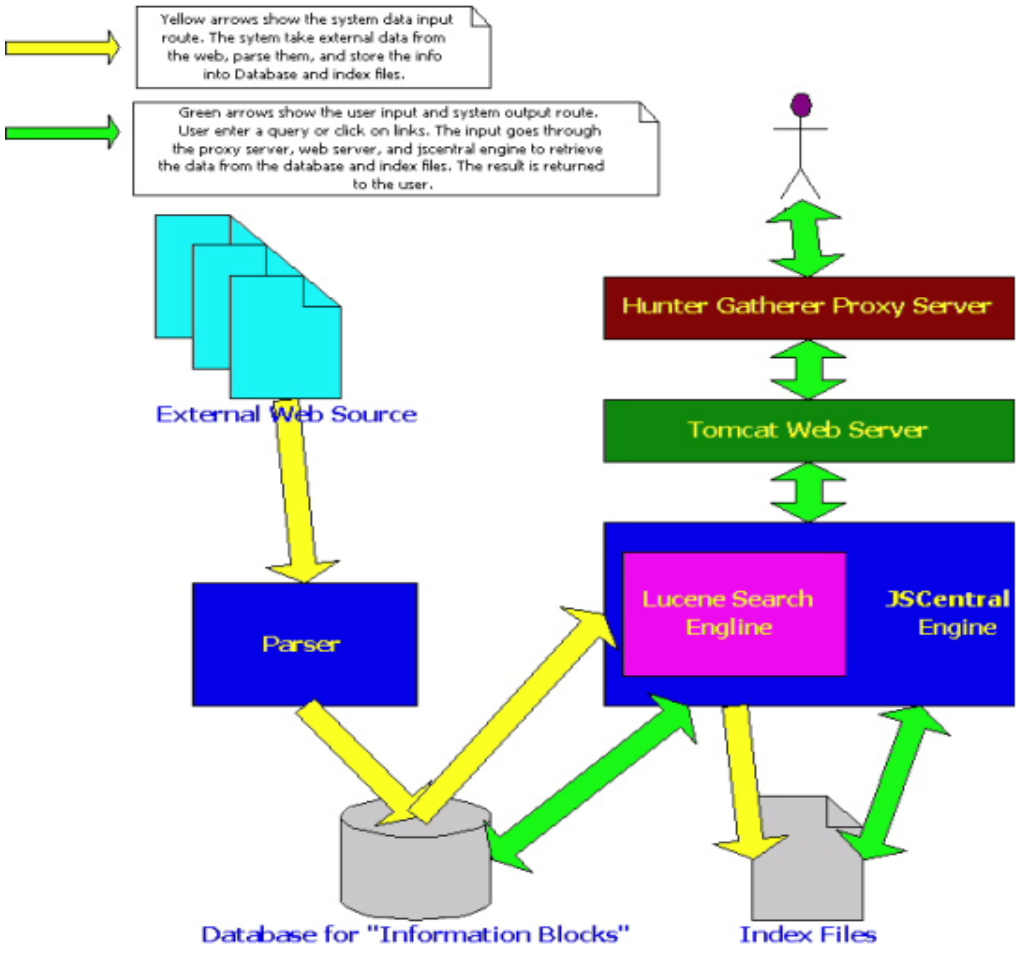

Fig. 8. Architecture diagram

The architecture is designed around the concept of (a) making data available to the interface as information blocks and (b) supporting strong interlinking among components. As Figure 8 above indicates, the current architecture reflects a largely two-stage process: preprocessing of web data for storage into a local database; post processing requests of that data for display. In the first stage, what we call our JSCentral engine gathers information from predetermined JavaScript Web resources, processes it (stores meta data like date, author, original source) into an *information block* a single, meaningful element of information. A block could be the answer to a question, a page of code that solves a problem or represents a function, a glossary description of an object. The engine then parses each block against recurrent key words to tag it for association with categories (a block can be associated with multiple categories). The blocks are then stored into our main database. The JSCentral Engine then parses and analyzes interactions by the user with the interface, and finds the best match among information blocks that relate to a particular selection. The search result is packaged into the specific format for its presentation in the interface. The presentation layer is where our main program logic lives. This component focuses on presenting the information retrieved from the database and processed by the JSCentral Engine to the user. The Hunter Gatherer component, which allows users to collect elements from within the domain, resides on top of the presentation layer. Its architecture is described in [12].

Currently we use the following technologies for our system: mySQL is our open source database solution and Java as the implementation language. We have reused considerable code from the parsing and term extraction components of the Apache Lucene Open Source project for the query interaction part of the service. We use Tomcat as our Web server + servlet/JSP engine. For the presentation layer, we use a combination of Java applets,

JavaScript, and HTML to show the content to the users. We use JavaScript and Frames in the presentation layer to implement the spatial view.

This is only a brief overview of the current architecture. For a detailed description of how the engine parses and categorizes information for presentation, please see [19]. We return to architecture issues in the Future Work section below.

5. Conclusions and Future Work

We have seen from our previous work, and the task analysis presented here that for some kinds of information problems, Google is not enough. Further, non-adaptive web sites, such as the various JavaScript sites, cost users time in learning the set up of the site and then how to navigate within this new space. To address these problems, we presented an approach that brings information in a domain into a coordinated spatial context and affords several interaction methods to support users interacting with the content such that they can adapt the content to support their perspectives. We use dimensional sorting in particular for this assist, so that users can reorganize the domain itself to support their perspective. This particular technique was developed as a direct outcome of our task analysis to support usersí current knowledge about a domain, and to endeavour to leverage that knowledge by letting the user see the domain oriented from that perspective. We have run preliminary reviews of our interaction design, but plan a more formal, large scale evaluation to help test the robustness of our current approach is planned. In the preliminary reviews, however, dimensional sorting was frequently deployed, and, consistent with previous evaluations, preview cues enhanced efficiency of path deliberations when navigating through the domain. Participants across expertise ranges reported a desire to have access to the prototype ìas soon as possible,î saying that one of the largest benefits was coordinating multiple resources into a single, associated domain.

The larger goal of the project is to define a general mSpace interaction model for domain access and exploration. If the tests are successful, we can anticipate several parts behind the interaction to which we will need to return. For instance, we do not currently rate information blocks for user level suitability. The current prototpye is an opportunity to determine if such labelling is required or beneficial. It may be enough for the user to see a component both to glean its appropriateness, and to benefit from knowing that that information exists, even if it is not wholly understandable. This seems to be what users are doing with the prototype. We also have yet to incorporate any kind of ranking of the information components. The question remains if, when the user can see much of this information in context, and assess its intelligibility to them directly, whether ranking responses to queries is a priority for effective support. We have predicated that mSpace provides better support information access and exploration; we also wish to investigate how mSpaces may afford knowledge building by supporting persistent context and association building. As indicated above, since the mSpace approach seeks to represent a domain in a structured way (similar to the way structured hypertexts represent a document [17]) we postulate that mSpace can be an effective front end for Semantic Web queries over a particular domain space [13].

Overall, the benefit of this user-determined, adaptable approach to the Adaptive Hypermedia community is to propose a suite of successful interaction approaches that have been shown both to support the tenets of AH (that there is value in adapting content for various users and contexts), but that their particular success is in giving the adaptive control to the user rather than the system. It may be that we have reached a place in AHís evolution where we might be able to create a taxonomy of kinds of information that are more appropriate to adaptable hypermedia and those kinds of discoveries that are more appropriate to adaptive, and, of course, where the twine might meet.

References

1. Brusilovsky, P. Methods and Techniques of Adaptive Hypermedia. User Modeling and User-Adapted Interaction, 6 (2-3). 87-129.
2. Chimera, R. and Shneiderman, B., An Exploratory Evaluation of Three Interfaces for Browsing Large Hierarchical Tables of Contents. In ACM Transactions on Information Systems (TOIS), (1994), 12.
3. De Bra, P. The Adaptive Web: Adaptive Educational Hypermedia on the Web Communications of the ACM: SPECIAL ISSUE: The adaptive web, May 2002, 60-61.
4. Diaper, D. (ed.), Task Analysis for Human Computer Interaction. Ellis Horwood, Chichester, 1989.
5. Neuwirth, C., Kaufer, D., Chimera, R., Gillespie, T., The Notes Program: A Hypertext Application for Writing from Source Texts, Proceeding of the Acm Conference on Hypertext. In Proceeding of the ACM conference on Hypertext, (November 1987), 121-141.
6. Plaisant, C., Shneiderman, B., Doan, K. and Bruns, T. Interface and Data Architecture for Query Preview in Networked Information Systems. ACM Transactions on Information Systems (TOIS), 17 (3). 320-341.
7. Schaffer, D., Zuo, Z., Greenberg, S., Bartram, L., Dill, J., Dubs, S., Roseman and M., N. Navigating Hierarchically Clustered Networks through Fisheye and Full-Zoom Methods. ACM TOCHI. 162 • 188.
8. Schraagen, J.M., Chipman, S.F. and Shalin, V.L. (eds.). Cognitive Task Analysis. Lawrence Erblaum Associates, Publishers, Mahwah, New Jersey; London, 2000.
9. schraefel, m.c., Contexts: Adaptable Hypermedia. In Adaptive Hypermedia and Adaptive Web-Based Systems International Conference, AH 2000, (Trento, Italy, 2000), Lecture Notes in Computer Science, 1892, 369-375.
10. schraefel, m.c. An Evaluation of Preview Cues for Exploration of Information Hierarchies. Dept. of Computer Science, University of Toronto, CSRG-462, 2002, 1-8.
11. schraefel, m.c., Karam, M. and Zhao, S., Preview Cues for Exploring Domain Hierarchies. In Interact 2003, (Switzerland, 2003, forthcoming).
12. schraefel, m.c., Zhu, Y., Modjeska, D., Wigdor, D. and Zhao, S., Hunter Gatherer: Interaction Support for the Creation and Management of within-Web-Page Collections. In Proceedings of the eleventh international conference on World Wide Web, (Hawaii, 2002), 172-181.
13. Shadbolt, N., schraefel, m.c., Gibbins, N. and Harris, S., Cs Aktive Space, or How We Stopped Worrying and Learned to Love the Semantic Web. In International Semantic Web Conference, (Florida, USA, 2003 in submission).
14. Shepherd, A. Task Analysis in Hci. in Monk, A.F. and Gilbert, N. eds. Perspectives on Hci: Diverse Approaches, Academic Press, London, 1995.
15. Shipman, F., Moore, J.M., Maloor, P., Hsieh, H., Akkapeddi, R., Spatial Hypertext: Semantics Happen: Knowledge Building in Spatial Hypertext. In Proceedings of the thirteenth conference on Hypertext and hypermedia, (June 2002), 25• 34.
16. Terry, M. and Mynatt, E.D., Side Views: Persistent, on-Demand Previews for Open-Ended Tasks. In UIST, (2002), 71 - 80
17. Wang, W. and Rada, R. Structured Hypertext with Domain Semantics. ACM Transactions on Information Systems (TOIS), 16 (4). 372 - 412.
18. Zellweger, P.T., Regli, S.H., Mackinlay, J.D. and Chang, B.-W., The Impact of Fluid Documents on Reading and Browsing: An Observational Study. In CHI 2000, (The Hague, Amsterdam, 2000), CHI Letters, 2, 249 - 256.
19. Zhao, S. and schraefel, m.c. Information Blocks: Working Architecture to Support the Mspace Prototype. Dept. of Computer Science, U of Toronto, CSRG-464, 2002, 1-7.