The xMem Project: Semantic Memory of Web Interactions

Stefano Ceri, Florian Daniel, Maristella Matera, Francesca Rizzo
Dipartimento di Elettronica e Informazione
Politecnico di Milano
Via Ponzio 35/a, 20133 Milano, Italy
{ceri,daniel,matera,rizzo}@elet.polimi.it

Abstract

Finding a previously visited page during a Web navigation is a very common and important kind of interaction. Although most commercial browsers incorporate history mechanisms, such mechanisms are typically very simple indexes of visited pages, sorted according to the time dimension. They are not very effective and are quite far from giving users the impression of a semantically aware, long-term memory, as it is available to the human brain. In particular they lack associative, semantic-based mechanisms that are essential for supporting information retrieval.

This paper introduces the xMem (Extended Memory Navigation) as a new method to access users' navigational history, based upon semantic-based and associative accesses. Its aim is to imitate some of the features of the human memory, so as to give users a better understanding of the context of their searches, by exploiting time-based ordering and semantic clues.

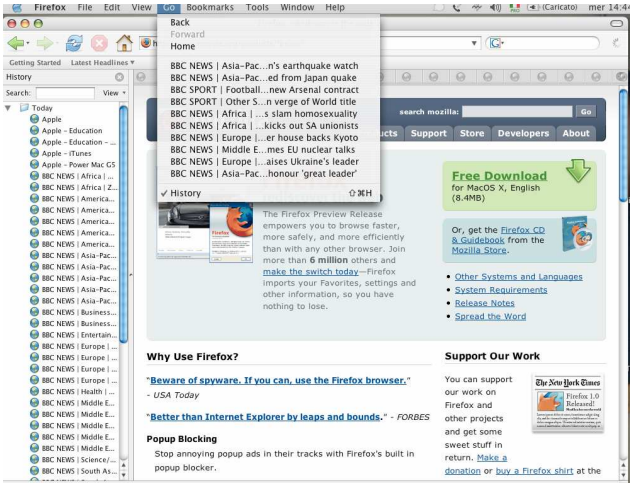

Figure 1. FireFox history mechanism. The "go" menu displays the last 10 visited URLs in a stack from the most recent to the oldest. FireFox also offers a history window that organizes information in chunks per days.

1. Introduction

As the amount of information on the World Wide Web continues to grow, efficient hypertext navigation mechanisms are becoming crucial. Among them, special attention must be devoted to effective history mechanisms enabling users to get back to information already met [18].

History tools are common components of Web site interfaces. They aim to support users to go back to the previously used information. Three factors motivate the introduction of these mechanisms in commercial browsers: (1) they can help users to navigate through the huge quantity of information provided by the Web, thus providing access to information previously visited; (2) they can substitute search engines for finding old pages and avoid the replication of navigations along intermediate pages to the destination one; (3) they can positively affect users' activities by reducing cognitive and physical navigation burdens: pages can be retrieved with little effort, and users can easily locate where they have been in the past.

Virtually, all commercial and experimental browsers provide users with methods to revisit pages of interest. We categorize them as *passive* and *active*. Passive history mechanisms (see Figure 1) are syntactic in nature, resulting from the navigation actions taken by the user. Active re-visitation mechanisms, such as bookmarks or the "back" button, have a more semantic quality and are explicitly created by users based on their interest level in the page [16].

Most of the results coming from the field of HCI [18, 12, 6, 13] show that more of the 30% of users' activities

on the Web are based on the use of the "back" button or of favorites, but they also show that reverse browsing mechanisms are time consuming and cognitively difficult to use, organize and envision. These reasons have lead us to introduce a new method to access navigational history.

This paper introduces the xMem (Extended Memory Navigation) project, which aims at providing advanced memory mechanisms to Web users, intermixing semantic aspects with the temporal dimension. The xMem solution requires the classification of information being shown on the Web according to predefined semantic categories, which are next used for indexing and retrieving. Such classification and indexing are effective for those Web applications that are previously described to xMem, and in particular those modeled at a high level of abstraction, by means of primitives of a conceptual model. However, the xMem architecture presented in this paper is quite general and applies as well to arbitrary Web pages, provided that these pages being browsed by users are also analyzed in parallel by the xMem tool by means of suitable wrappers.

The paper is structured as follows. Section 2 describes the main idea underlying xMem and outlines the goals of our research. Section 3 then introduces the functional architecture of xMem, while Section 4 discusses some issues concerning design and implementation of the current prototype tool. Section 5 provides insight into related work and describes already collected experiences within the context of history mechanisms in the HCI field. Finally Section 6 presents some conclusions and gives an outlook over ongoing and future work.

2. The *xMem* Web Interaction Memory

The xMem project offers to users a specialized Web site hosting a repository containing the individual user's memory. The key idea of xMem is the "transparent" transmission to the memory Web site of the URLs of the pages being browsed by the users. Then, suitable rules extract the information from the URLs and populate the memory with semantic content. In the light of the previous considerations, the aim of our work becomes twofold:

(i) Providing easier and more intuitive navigation-history retrieving mechanisms. Web navigation can be aided by novel navigation history tools providing enriched memory access mechanisms that build on semantics-based search criterions categorizing available pieces of information. The xMem approach further puts special focus on accessing the long-term memory and thus fosters the active use of history data.

(ii) Fostering ubiquitous accessibility of the navigation history by making it accessible over the Internet. The gathered and classified information must be always reachable in order to become an active help facility that can be integrated within the user's browsing environment.

In order to provide users with some added value (with respect to usual history mechanisms), besides chronologically ordered lists of URLs, xMem also supports further semantics in presenting history data. So-called semantic classes or categories are associated to groups of URLs in order to provide high-level meanings for the contents of the related Web sites. For associating URLs and categories, proper filters can be defined, which allow specifying syntactic rules describing relationships between specific URL structures and categories.

xMem also provides a mechanism for automatically deriving semantic classes starting from application specifications expressed through WebML conceptual models, and a set of generic filters that apply to any WebML application. WebML is a conceptual modeling language that provides a set of primitives for the design of information content and hypertext front-end of Web applications [17, 7].

Rules are particularly powerful when the pages being browsed by the user have a known design, because the content can simply be derived from the URL; when instead the design is not known, pages can be re-materialized and then analyzed by suitable rules, which are capable of detecting the main concepts being displayed by the pages themselves. In any case, xMem maintains remote log data (with respect to users and 3rd party Web servers) about navigation actions of users, by tracking the requested URLs. We adopt an online logging mechanism that allows accessing history data, classifying navigated contents and calculating related statistics at runtime.

3. *xMem* Architecture

The software architecture of the xMem tool prototype is primarily influenced by two goals of our approach: (i) adopting remote logging mechanisms for (ii) providing online access to logged data. Remote logging builds on the client-server paradigm, online log access suggests splitting the overall tool into two logical components, one for each communication direction. Figure 2 graphically depicts the resulting functional architecture, roughly divided into *Client PC*, *3rd party Web applications* and *xMem Server*.

Users navigate the Internet by means of a common Web browser by continuously requesting pages of Web sites or applications residing on 3rd party Web servers. In order to take advantage of the semantic navigation history facilities of xMem, users must register themselves as new xMem users and install the so-called xMem *Tracker Client* on their client PCs. This component is in charge of monitoring user navigation actions and communicating them to the xMem *Tracker Server*, which is responsible for feeding the incoming messages into the *URL Repository*. The *Classifier* component parses syntactic filters with respect to logged page

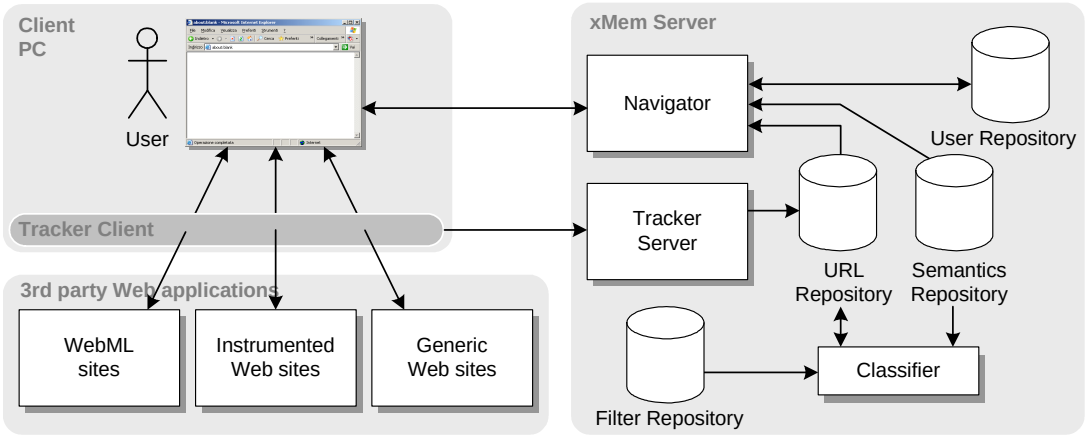

Figure 2. xMem functional Architecture.

requests and associates them to semantic classes stored in the *Semantics Repository*. By means of the so-called *xMem Navigator* Web application, users can then access their personalized navigation history over the Internet and interactively browse logged data at a comprehensible level of abstraction.

The box 3rd party Web applications is divided into *Model-based Web sites*, *Instrumented Web sites* and *Generic Web sites* for representing the different "semantic expressiveness" they enclose with respect to the xMem infrastructure. We will deepen this aspect later on.

3.1. Data Repositories

The xMem project consists of several components that share the same data source. The implementation of the correspondent database depends on the expected load at runtime and can consist either in a single database on the xMem server itself or in a freely distributed server architecture. However, at a conceptual level, we distinguish mainly four logical "repositories", each dedicated to collect functionally similar data:

- **User Repository**: gathers user profiles and preferences for registered xMem users and divide users into different user groups for managing access rights. These data are the basis for personalizing contents and history data and allow fulfilling different functional requirements with respect to user groups. The User Repository must implement at least the following user groups: *xMem User*, *Site Developer* and *Administrator*. xMem users are those that use the history tool, developers are those providing xMem support for their Web sites and administrators manage the overall xMem tool.

- **URL Repository**: contains the actual log data for each registered user; visited Web pages are stored as URL strings. The xMem Tracker accesses the URL Repository for adding HTTP requests, but also Navigator and Classifier make active use of it. The URL Repository masters the highest workload.

- **Semantics Repository**: collects the so-called semantic classes, which provide high-level meanings for classifying the contents of Web pages. Such classes can be defined either automatically or manually by 3rd party site designers. Each adhering Web application is provided with its own set of classes. For WebML applications, a subset of the application's schema is stored within the Semantics Repository for deriving semantically relevant concepts.

- **Filter repository**: contains the set of filters that allow matching URLs and semantic classes by means of syntactic similarities. Each application has its own set of filters. A default set of filters applying to WebML applications is provided by xMem, for all other applications filters can be specified manually or automatically, as explained in section 3.6.

3.2. The xMem Tracker

The history tracking mechanism of xMem comprises two components, one residing on the *xMem Server*, the other running (as background process) on the client PC. At each navigation action, the *Tracker Client* automatically sends a message to its server counterpart communicating xMem user identifier, and requested HTTP URL. The *Tracker Server* inserts the request into the underlying URL Repository, which terminates the tracking process for that URL.

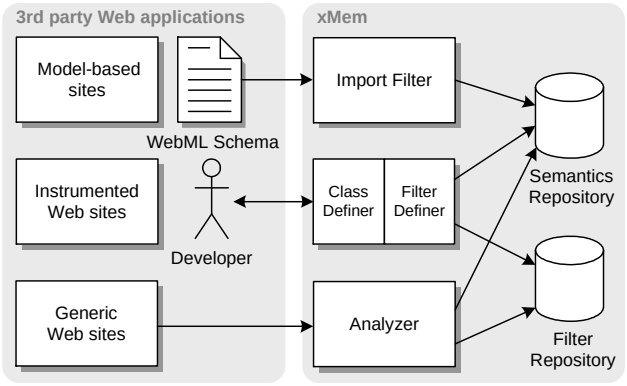

Figure 3. Defining semantic classes for Web sites.

3.3. The xMem Navigator

The interactive component of xMem, the *Navigator*, is a Web application developed through WebML; as such, it is accessible over the Internet, and can be deployed on arbitrary Web servers. Hence, for accessing their history data, users do not need to get used to new interaction paradigms.

The Navigator provides the interface toward history data. It allows users to choose between several mechanisms for accessing navigated pages. Besides chronological and alphabetical lists of URLs, it also shows a hierarchical list of the concepts (semantic classes) that summarize the contents visited by a user. The finest granularity is given by the tracked URLs themselves and allows users to comfortably recall the actual page searched. Also, the Navigator allows users to modify their user profiles.

3.4. The Classifier

One of the core components of xMem consists in the *Classifier*. Its inputs are the tracked URLs and the set of filters of the respective Web application. The Classifier runs those filters over the tracked URLs for constructing semantic associations between *URL Repository* and *Semantics Repository*. It therefore acts as an interpreter of syntactic filters, according to their definition in section 3.6.

3.5. Defining Semantics for Web Pages

For providing users with semantic clues about their past navigation actions, xMem makes use of collection of semantic classes contained in the *Semantics Repository* and describing the navigated pages by means of a natural language, easily comprehensible by users. In this section we show how semantic classes are specified in xMem during subscription of adhering applications.

Subscription in this context means providing, through an appropriate user interface for Web designers, the description of core pieces of information about the application to be added: a unique base URL identifying the application, a set of semantic classes in an automated or manual way, and the qualifying filters for each class. Designers need to subscribe their Web sites in order to gather full xMem support.

For the purpose of defining semantic classes, we distinguish several kinds of Web site implementations. Web applications can be developed using conceptual models, such as WebML, and even automatically generated [19], or they may be freely hand-coded without any particular design method. The former are characterized by a certain degree of already intrinsically modeled high-level semantics, whereas the latter do not provide any additional information besides plain HTML code. Thus, according to the availability of additional semantics and the way this can be fed into xMem, we distinguish three families of Web sites (see Figure 3):

- **Model-based sites**: are those designed by means of WebML and implemented according to WebML design rules. Intrinsically, the WebML design primitives already contain additional information with respect to plain HTML (e.g. the name of content units within pages, such as *Book Details* or *Author*) that can be fed into the Semantics Repository and thus can be interpreted as semantic classes. In addition to the semantics provided by the schemas, WebML applications are characterized by coherent URL formats throughout the whole application that easily can be translated into syntactic filters, which can be generalized and applied to any WebML application. Starting from WebML schemas, it is thus possible to provide a basic set of universal filters (permanently stored within the *Filter Repository*) and to automatically extract semantic classes associated to them. In Figure 3 this process is expressed by means of the *Import Filter* block, which takes in input a WebML schema and provides in output the set of semantic classes describing the site's contents.

- **Instrumented Web sites**: are those sites that do not rely on WebML schemas or other modeling languages, but whose developers adhere to the xMem project anyway. For this purpose, xMem provides the *Class Definer* and *Filter Definer* tools, which allow providing the additionally needed semantics manually by means of proper semantic classes and specifying a set of filters matching the peculiarities of the Web site's URL encoding conventions. Depending on the conventions adopted within the site, filters may differ in their complexity. Hence, the full features of xMem can also be exploited by any Web site whose designer adheres to xMem.

Target	Type	Condition
Schema	string	contains, doesn't contain
User-Info	string	contains, doesn't contain
Host	string	contains, doesn't contain
Port	int	equals, differs from
Path	string	contains, doesn't contain
Query	string	num. parameters, exists parameter
Fragment	string	exists, contains, doesn't contain
Parameter	string, float*	contains, doesn't contain, equals, differs from, greater than, less than

* Depending on the adopted condition expression, parameter types are assumed being either string or float.

Table 1. Target components and available conditions.

- **Generic Web sites**: are those applications that do not know about the xMem project and thus neither provide WebML schemas nor other forms of additional information that could help classifying recorded navigations. Tracked URLs of this category are rematerialized by a special *Analyzer* module (cf. Figure 3), which parses the respective HTML encoding for automatically extracting significant "keyworks" summarizing the contained contents as accurate as possible. Extracted keywords represent *semantic classes*, and are associated to the analyzed URL by means of proper filters, as well generated automatically by the Analyzer. The keyword extraction algorithm developed so far is still based on pure syntactic heuristics; semantic or ontology-based ones are under investigation and promise to enhance the accuracy of the keyword extraction algorithm on the one hand, and to provide means for clustering and categorizing keywords on the other one.

Although a good design of Web sites should allow covering with proper semantic classes most of their pages and contents, we do not believe in the usefulness of a complete coverage of all pages and contents by means of semantic classes. A semantic classification of error pages, for example, would be rather confusing to users. Therefore, even within a subscribed application, there may be tracked pages without any further associated semantics.

3.6. xMem Filters

According to their scope within xMem, we distinguish two kinds of filters, global filters and local filters. *Global*

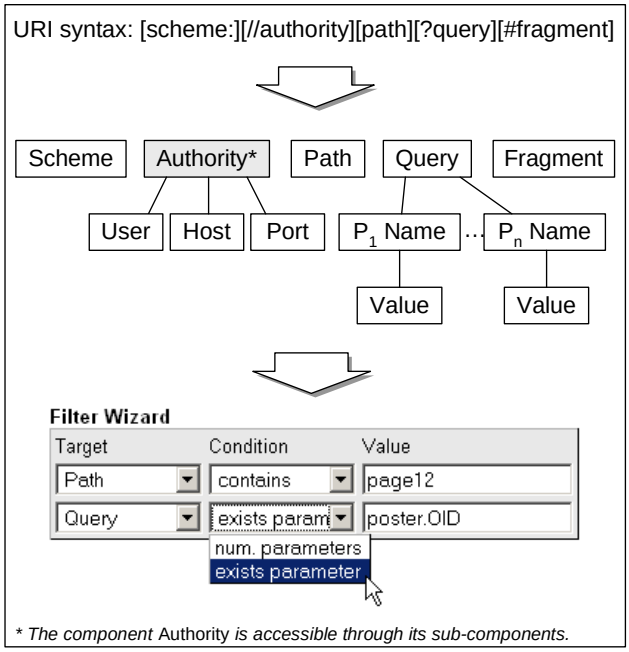

Figure 4. Decomposing URL strings and defining rules over its components.

filters are built upon the URLs of applications and check whether incoming URLs belong to one of the subscribed applications. Their scope is global within xMem and thus they are applied to all tracked requests. Suitable global filters are automatically created during the subscription of an application.

Conversely, *local filters* are only applied, once a URL has been associated to a particular application. The mechanisms mentioned in the previous section are adopted for deriving semantic classes covering the significant pages of an application. For each of these classes proper filters must be specified. Due to the fact that more different URL structures could lead to the same concept, several filters may be defined for the same class. At least one of them must match a particular URL in order to associate URL and class.

Filters consist of one or more rules and one action. For executing the action, which consists in associating URLs and semantic classes, all rules of the filter must evaluate to true. A rule is composed of target, condition and value; Table 1 shows the possible instances for the first two, the *value* is provided by designers (cf. Figure 4). Target elements are derived from URLs according to [4], which classifies URLs registered by the *xMem Tracker* as absolute, hierarchical and server-based URIs (Uniform Resource Identifiers). Figure 4 graphically illustrates the applied decomposition of URLs and also shows how, based on the resulting atomic components, rules are defined by means of a proper Filter

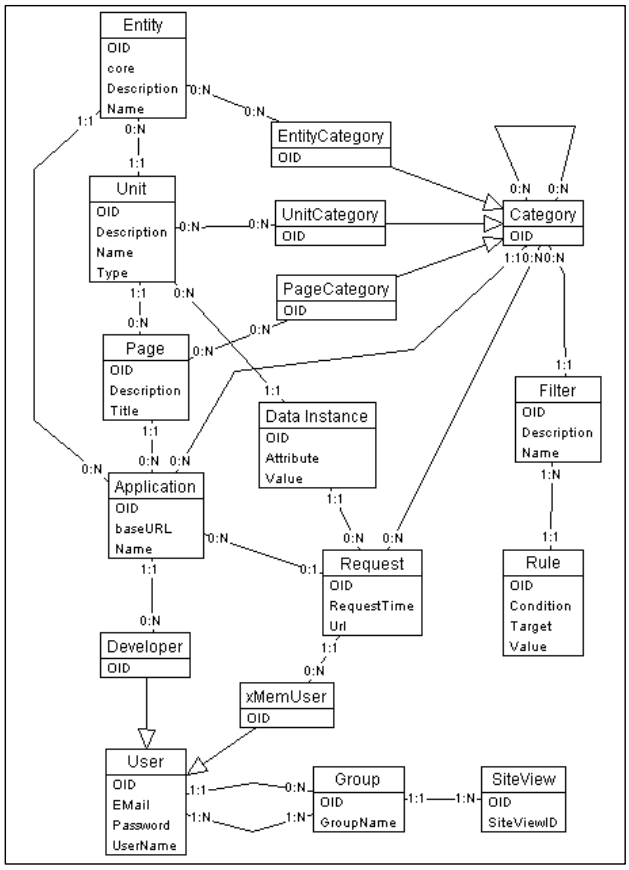

Figure 5. The xMem prototype data source.

Wizard. The *Filter Wizard* provides an interactive user interface for defining rules for filters. The so collected user inputs are translated at runtime into suitable regular expressions, one for each rule of the filter, and evaluated for the respective target component.

4. Prototype

We are currently working on a prototype tool for assessing the usability and usefulness of the xMem concepts. The following subsections provide insight into the state of the art of xMem. For presentation purposes, the proposed Web interface and the structure of the underlying data source are a simplification of the actual prototype.

4.1. Data Model

Figure 5 shows the Entity-Relationship diagram of the prototype data source containing the repositories outlined in section 3.1 within a single database. The database is shared among the various xMem components, but its design is particularly tailored to the requirements of the *Navigator* WebML application.

Entities and repositories can be matched as follows: The entities *User*, *Group* and *SiteView* implement the user model within WebML applications [17] and thus respond to the needs of the *User Repository*. The entities *xMemUser* and *Developer* further refine the repository, representing specific properties of the two user classes. The *URL Repository* can be associated to the entity *Request*, that collects navigated pages and request times. In addition, in case of WebML applications, also references to the specific data instances visited can be stored (entity *Data Instance*). The *Semantics Repository* comprises the entities *Application*, *Entity*, *Page* and *Unit* for collecting the data extracted from submitted WebML schemas. The entities *Category*, *EntityCategory*, *UnitCategory* and *PageCategory* then build the proper *Semantics Repository*. Finally, the *Filter Repository* consists of the entities *Filter* and *Rule*. A filter can have one ore more rules and is associated to its specific category. The rule comprises the attributes *Target*, *Condition* and *Value*, which reflect the rule structure introduced in section 3.6. Each application has its own set of categories and thus its own set of associated filters.

4.2. The xMem Tracker

The Tracker is a small Java HTTP sniffer program to be installed on the client PC. Once launched, it operates in a completely transparent manner in the background. Users have the option to turn the tracker on and off, thus activating and de-activating xMem. While running, it is accessible by means of a small tray icon in the operating systems' taskbar. A mouse click on the icon opens the context menu and provides the user with facilities for managing his authentication data and for starting and stopping the tracking service. No data is sent without the explicit permission of the user. Tracked page requests and user identifier are encoded and sent to the xMem Server via the Internet as common HTTP requests.

The Tracker Server consists of a single Java servlet that listens for incoming messages and, after successfully authenticating the user, adds the request properly formatted to the user's log data.

4.3. The xMem Navigator

Figure 6 gives an idea of the Web interface of xMem and outlines the basic functionalities offered by the Navigator prototype. Two landmark areas (reachable through the application's main menu) are available, one concerning user profile management, the other providing proper history browsing facilities. The home page *User Details* of the site view (labeled with an "H") also is the default page

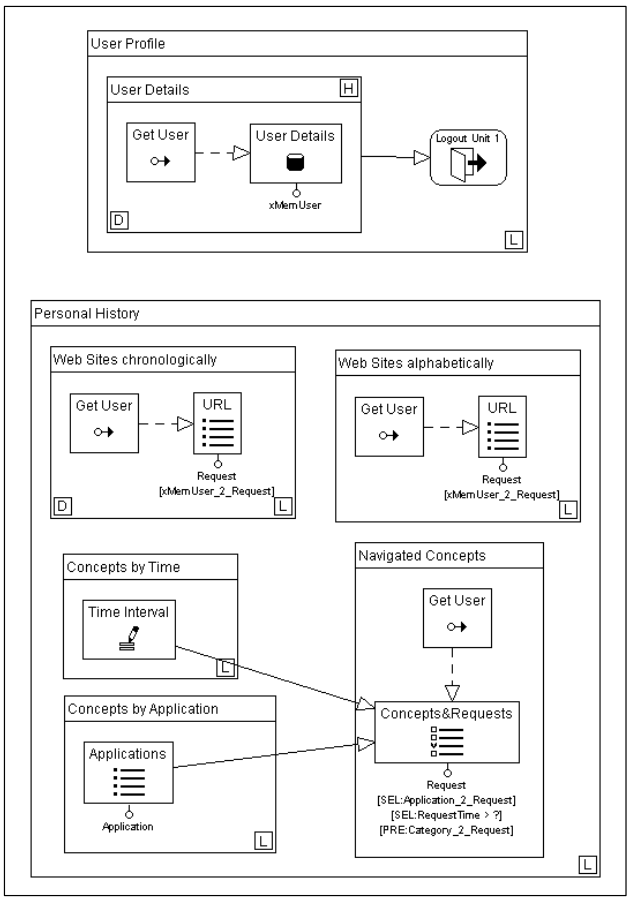

Figure 6. WebML schema for the site view of xMem users.

(label "D") of the *User Profile* area; thus it is presented to users as first page once they enter the area. The area *Personal History* has as default page the page *Web Sites chronologically*. That page and the page *Web Sites alphabetically* respectively show chronological and alphabetical lists of navigated requests, personalized with respect to the current user. The page *Navigated Concepts*, at the other hand, exploits the additional semantics stored in the data source in form of different categories and associated by the wrapper mechanism to the visited pages. This results in a page that provides a hierarchical list (in form of a so- called *multidata unit*) of navigated Web sites structured by meaningful categories. From this point on, the user can browse the concepts according to several criteria (similar concepts, concepts seen during previous or subsequent visits, etc.).

Once a user has successfully retrieved the concept he was searching for, the leaves of the hierarchical list show all URLs that match the interactively specified access conditions. By clicking on one of the URLs the user is forwarded to the actual page (within a new browser window).

4.4. The xMem Administration Tools

Besides the site view for xMem users, the Navigator also provides proper site views for 3rd party application designers, administrators and a public site view for logging in. The administrator's site view serves mainly the purpose of managing users, groups and applications. The designer's site view allows subscribing Web applications, being they WebML applications or not. A detailed presentation of this two site vies is out of the scope of this paper; however, section 3.5 and Figure 3 provide little insight.

4.5. The Classifier

The Classifier is currently implemented as integrated component of the Tracker Server. The URL decomposition mechanism is based on the two Java classes `URL` and `URI`, whereas the filter parsing mechanism makes extensive use of the Java `regex` package. Incoming URLs are instantly categorized and stored into the database, by extracting semantic information such as entity names or object instance identities from them.

5. Related Work

Most commercial browsers incorporate history mechanisms. However, users still do not extensively use them. Catledge and Pitkow [6] state that these complex history mechanisms are often not used; and the 40.6% of the user actions involve the browser's back button. In [12] results show that "0,1% of page accesses were through the history list, 42% of page accesses used the back button" for all of the pages navigated by subjects.

These and many other findings suggest that current browsers lack an efficient method for revisiting Web pages, and that "current interface for browsing on the WWW are still primitive... They do not aid users in accessing already visited pages without much cognitive demand" [13].

History mechanisms can be distinguished in *passive* and *active*. The first class works by maintaining some kind of information about a particular Web session. The browser stores some subset of the pages visited by the user, typically in a list. However, since most history mechanisms store only the links on the last spoke traversed (i.e., the current path in a depth-first traversal), by using this mechanism the user may only reach a small portion of previously visited pages. If, when deep in a search tree, the user finds an interesting page (i.e., one to which she/he may wish to return), it may be difficult to preserve that link while continuing with the search.

Active mechanisms, such as bookmarks, provide users with a method to mark interesting Web pages they would

like to return to. For most people, however, bookmarks create a new difficulty: as users find a large number of pages interesting, bookmark lists quickly become long and unwieldy. Although most browsers provide mechanisms for editing and maintaining bookmark lists, they are not an ideal mechanism for maintaining context within a single browsing session. In general, it seems that active history mechanisms such as the back button and bookmarks function are more effective to support short and long term information memories retrieval than what the passive mechanisms do. In fact, if a user is navigating a Web site it is more easy to him to go back to the pages he wants to revisit using the back button then opening the history mechanism window, analyzing the information displayed as URL, choosing one among them and click on it.

On the other side, users experience frustration in retrieving already visited pages when a certain amount of time is passed from the first visit. The reason is that the context in which that page was being viewed is lost. The path-following method (on which the most part of the history mechanisms are based) for retrieving long term history memories imposes users to traverse in reverse order their previously visited pages. This method relies on users remembering their navigational behaviors, either because they must recall the page visited and their sequence or because they must realized that they can return to a page by retracing a particular pathway. Indeed, the lack of efficient mechanisms for maintaining context may be partially responsible for this phenomenon.

Few researchers have considered new metaphors for browsing and collecting information on the Web; we next briefly describe their work. Most of the methods introduced in the following use a mix of graphical representations and a sense of context. Some of them place on the end users an extra request of cognitive effort. Most of them are features of experimental browsers and are only able to show syntactic information, not semantic information.

IBM's Web Browser Intelligence (WBI) tool [1] is a browser companion with personal history functions intended to make Web browsing more efficient by annotating hyperlinks on all Web pages with traffic signals, and this performs well for functions such as: remembering visited pages, providing a keyword search through the text of pages already visited, noticing patterns in the Web browsing behavior and suggesting shortcuts, and automatically checking favorite Web pages for change.

Some types of systems present graphical maps through which a user may navigate. Some are like HTML frames in that they are created once by the content provider. Others are created on the fly by the Web browser. For example, WebTOC [15] is an automated system for creating table of contents (TOC) frames for sets of Web pages. The TOC frame can be quite useful, and having that frame actually run a Java program allows it to more dynamically present the desired information. Its main drawback is that it occupies a large portion of the screen. WebNet [8] is a browser extension that displays a graphical representation of the hyperspace being explored. It does so dynamically, and independent of the content provider. In fact it can do so across many sites. It is a challenge, however, to present the graph in such a way that the contextual information is highlighted.

DeckScape [5] is an experimental browser based on the concept of deck. Each deck is a linear stack of pages that the user can leaf through. As with history mechanisms, if a user starts from page A and goes to B, B is added to that stack or deck. However, unlike history mechanisms, if the user goes back to A and then traverses a link to C, B is not lost; it remains in the deck of pages. Also, the user can always revisit any page since no page is ever lost. However, users are cognitively loaded with the responsibility of maintaining pages logically in different stacks unless decks are pruned regularly.

Elastic Windows [13] also provides a mechanism that illustrates graphically the hyperspace in which a user is navigating, but it does so more interactively. If the user selects a link using this system, the contents of the corresponding page do not replace the currently displayed page; instead, the new page is displayed alongside its parent. Selecting multiple links from a page results in all the new pages being displayed alongside the parent, but in a smaller size. The same operations may be performed on any window in the browser. Users are provided with functionality to manage the windows by collapsing some sections of the hierarchy, while maximizing the size of others. Since the complete hierarchy is visible at anytime, users can easily move in the hierarchy while not losing their search context. The simultaneous display of multiple pages again places a management burden on the user; screen real estate can easily become cluttered if not managed efficiently. Cyclic links are not dealt with very well; the same page may be displayed multiple times, further wasting screen real estate. Elastic windows enables users to look at many Web pages at once, whereas a breadth first navigation technique provides a linear order for web visitations.

Scratchpad is a prototype incorporated in Chimera 1.70 [14], an experimental browser. Scratchpad introduces two mechanisms to support web navigation: a re-visitation mechanism called Dogears serves as a set of temporary bookmarks, and a breadth-first traversal mechanism is adopted as alternative navigation strategy to depth-first traversal. Scratchpad is not a history mechanism, but a navigational planning tool, applied to support users' visits of new pages. The information gathered by the breadth-first search and regarding possible future page visitations opens new ways for intelligent pre-fetching of those pages, thereby allowing faster navigation. Both mechanisms may

be added to existing browser technologies [16].

More recently [11] the Microsoft research center has worked on the system *tuff Ive Seen* (SIS), which is able to support users in retrieving and reuse information already seen locally on the PC. The system aim at facilitating the information seeking behavior by providing an index of information that a user has seen (email, web page, document, appointment) and, in addition a set of rich contextual cues about the searched information, made available from the previous accesses.

All solutions described above respond to the need to record users' navigational history (URL lists) for allowing the successive re- access of visited pages. xMem works beyond these mechanisms analyzing and interpreting the structure of recorded URLs and making available the results of this process to end users. This is obtained independently from the browser in use.

6. Conclusions

We have argued that current browsing functionalities do not adequately support retrieving information from a user's navigation history. xMem improves passive history mechanisms by making use of new semantic criteria to organize and show the navigational history instead of simply exploiting time-sorted history mechanisms that prevail today. This retrieving strategy makes navigation to already visited information easier and more effective because it provides a much richer notion of semantic context in which information has been seen. Context-dependent history mechanisms sustain users' episodic memory of the visited web pages.

Further investigations are necessary to refine our prototype, especially for what concerns the analysis of generic Web applications. We are currently studying wrapping technologies such as LIXTO [2, 3] and RoadRunner [9, 10] to see if their functionalities apply to the xMem context. We will also extensively experiment the use of the prototype, so as to assess its usefulness and usability. We also are interested to gain insight into the aspects of the history mechanisms that are most frustrating to users in an effort to suggest improvements for such features for future implementations.

References

[1] R. Barrett, P. Maglio, and D. Kellem. Web browser intelligence: Opening up the web. In *Proc. of COMPCON'97*, 1997.

[2] S. Baumgartner, S. Flesca, and G. Gottlob. Supervised wrapper generation with lixto. In *Proc. of VLDB'01, Rome, Italy*, 2001.

[3] S. Baumgartner, S. Flesca, and G. Gottlob. Visual web information extraction with lixto. In *Proc. of VLDB'01, Rome, Italy*, 2001.

[4] T. Berners-Lee. Uniform resource identifiers (URI): Generic syntax. RFC 2396, 1998.

[5] M. Brown and R. Shillner. Deckscape: An experimental web browser. Proc. of WWW'95: Technology, Tools and Applications, April 1995.

[6] L. Catledge and J. Pitkow. Characterizing browsing strategies in the world-wide web. Proc. of WWW'95: Technology, Tools and Applications, April 1995.

[7] S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Designing Data-Intensive Web Applications*. Morgan Kauffmann, 2002.

[8] A. Cockburn and S. Jones. Which way now? analysing and easing inadequacies in www navigation. *Int. J. of Human-Computer Studies*, 45(1):105–129, July 1996.

[9] V. Crescenzi, G. Mecca, and P. Merialdo. Automatic web information extraction in the roadrunner system. Proc. of the DASWIS'01 ER Workshop, 2001.

[10] V. Crescenzi, G. Mecca, and P. Merialdo. Wrapper-oriented classification of web pages. Proc. of ACM SAC'02), 2002.

[11] S. Dumais, E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, and D. Robbins. Stuff ive seen: A system for personal information retrieval and re-use. In *Proc. of SIGIR03, July 28 August 1, 2003, Toronto, Canada*, 2003.

[12] GVU's WWW Surveying Team. Gvu's 10th www user survey, 1998.

[13] E. Kandogan and B. Shnaiderman. Elastic window: A hierarchical multi-window world wide web browser. In *Proceedings of UIST'97*. ACM, 1997.

[14] J. Kilburg, G. Niklasch, and W. Edmond. Chimera-1.70, x11/athena world-wide web client. http://www.unlv.edu/chimera.

[15] D. Nation, C. Plaisant, G. Marchionini, and A. Komlodi. Visualizing websites using a hierarchical table of contents browser: Webtoc. Proc. of Human Factors and the Web'97, Denver, June 1997.

[16] D. Newfield, B. Sethi, and K. Rayll. Scratchpad: Mechanisms for better navigation in directed web serarching. In *Proc. of UIST'98*. ACM, 1998.

[17] Politecnico di Milano. The web modeling language. http://www.webml.org.

[18] L. Tauscher and S. Greenberg. How people revisit web pages: Empirical findings and implications for the design of history systems. *Int. J. Human Computer Studies*, 1997.

[19] WebModels. Webratio site development studio. http://www.webratio.com.