Extended link visualization with DHTML: The Web as an open hypermedia system

Report

Author(s):
Oberholzer, Glenn; Wilde, Erik

Publication date:
2002-01

Permanent link:
https://doi.org/10.3929/ethz-a-004284580

Rights / license:
In Copyright - Non-Commercial Use Permitted

Originally published in:
TIK Report 125

Extended Link Visualization with DHTML: The Web as an Open Hypermedia System

Glenn Oberholzer and Erik Wilde
Computer Engineering and Networks Laboratory
Swiss Federal Institute of Technology, Zürich

TIK Report 125
January 2002

Abstract

The World Wide Web is by far the most successful hypermedia system, its name often being used synonymously for the Internet. However, it is based on a rather restricted hypermedia model with limited linking functionality. Even though underlying systems may provide a richer data model, there is still the question of how to present this information in a Web-based interface in an easily understandable way. Assuming an underlying system similar to Topic Maps, which allows storing, managing, and categorizing meta data and links, we propose a presentation of extended links. We try to provide a usable way for users to handle the additional functionality. The mechanism is based on already available technologies like DHTML. It is one facet of our approach to make the Web more interconnected and to work towards a more richly and openly linked Web.

Keywords:

Electronic publishing (020), Graphic design (029), Hypermedia (036), Internet (045), World Wide Web (084), XLink, Linkbases, DHTML

1 Introduction

Compared to many other hypermedia systems [24,11,16], the linking capabilities of the World Wide Web are rather limited. It only makes use of a few concepts of hypermedia. In recent years, however, new recommendations issued by the W3C like XML [3], XLink [10], and XPointer [9], or ISO/IEC's Topic Maps [19] have tried to overcome this shortcoming. Due to the popularity of the Web, efforts have to be made to integrate and improve the current system with more sophisticated hypermedia concepts. More sophisticated hypermedia systems are in great demand, due to the growing amount of information provided by the Internet, and thus the growing need to be able to navigate the Web more easily.

This paper gives a short introduction of extended link functionalities and discusses possible ways of displaying these augmented links to the user. The paper primarily focuses on a solution that is already feasible with today's technology using DHTML.

An HTML link is defined as follows [29]: "A link is the connection from one Web resource to another." The main implications of that definition are that

- a link has exactly one source and one destination,
- its activation (for example by clicking on its graphical representation) initiates a traversal at the source which leads to the destination, and
- it establishes a relationship between the source and the destination.

Additionally, an HTML link is always embedded into the document, it is explicit and pre-computed [1]. Hypermedia research however suggests broader and more general link functionality. Namely n-ary links (linking more than two locations), separation of link and content, and annotations, are mentioned and implemented in existing systems [16,21,7,5,2], as described in Section 2. A more elaborate link model would not only increase the capabilities of the Web as a hypermedia system, the usability of the Web would also be improved. The main improvements from a user point of view are:

- *Educated choices*

  Right now, users are left with little or no feedback as to where a link is taking them. Including annotations and/or showing the type of relationship with destination resources enables the user to make an informed decision.

- *More and better choices*

  Multi-ended n-ary links give users the choice of where they want to go, as opposed to a single destination. This reduces the amount of time and traffic caused by searching through unrelated information.

- *Link quality*

  When links are stored separately from the content document, they can be more easily maintained and controlled. Broken links, a common problem in today's fast changing Web, can be avoided more easily.

- *3rd party links*

  Links which are maintained separately may be used to associate information in a way which would probably not be supported by the content's original provider, such as linking statements in product descriptions with resources containing critical remarks about these statements.

The very flexible and powerful XML framework provides the foundation for several initiatives which are trying to implement the above mentioned improved functionality. XLink [10] and XPointer [9] are W3C specifications that enable a better way of linking XML content.

XLink extends HTML linking functionality and first efforts have been made to provide a framework on how to apply styles to XLinks [33]. As of today, however, major browsers do not support XLink properly (only a small subset of XLink, the *simple links*, which only provide HTML-like linking features, are supported in the newest revisions of the major browsers). Therefore, displaying extended links in regular browsers today requires the help of existing technologies like DHTML. However, the concepts discussed in this paper can also be implemented in XLink-enabled browsers in the future.

In order to implement and manage the new link types, several approaches can be used. One of them might be a data model like XLinkbase. The XLinkbase model will be explained in more detail in Section 3 and is used in examples throughout this paper.

In terms of visualizing the new link concept, the authors have come up with an interface using standard components and technology of a Web browser and reusing known GUI widgets. This is important, because the key to get the user to use a new functionality is acceptance, which is achieved by providing good usability [26]. Additionally, learnability as one of the important components of usability [25] is greatly improved by presenting known concepts to the user.

Related work on hypermedia systems and visualizations on the Web are discussed in the following section. After explaining the philosophy behind XLinkbase as a possible underlying hypermedia system, the paper introduces the rationale used in the implementation of the prototypical Web interface. Section 5 discusses the actual implementation and technical details. The paper concludes by addressing open issues and possible future projects.

2 Related Work

Various open hypermedia systems have been adapted to the World Wide Web. Following, a number of systems that bring together *Open Hypermedia System (OHS)* [27] concepts and the Web are presented. The focus is put primarily on the way links are represented on the Web. While the selection certainly is not exhaustive, it showcases important initiatives.

Grønbæk [14] suggests a format called *Open Hypermedia Interchange Format (OHIF)*, which is similar to XLink, but features a richer data model. The Webvise interface and WebDAV server allow the publishing of OHIF sites. The Webvise client extracts the meta data stored in a document and displays it in a separate application window. Alternatively, a proxy server can be used to include additional links and display annotations in a JavaScript pop-up [15]. This specific service has been discontinued. Multi-ended links are possible in the system, yet the Webvise client is needed.

HyperWave [21] is a second generation Web system enabling a more structured organization and visualization of Web content. However, there is no direct way to create multi-ended links. The interface is transparent to the user. The user does not realize that the underlying system is HyperWave; the presentation of links is in traditional HTML syntax.

Both previously described systems store their links separate from the content. So does Webcosm [8, 16], which has been developed at the University of Southampton and is now commercialized in *Portal Maximizer*[1]. Portal Maximizer organizes information in "themes". It also injects external links into documents when they are requested. The one-to-many functionality of links is implemented by showing a list of links on a new page when clicking on a link (in our opinion a disruptive technique).

A commercial Topic Map visualization is implemented in *Ontopia*'s Topic Map Navigator[2]. The approach there is different however, as the navigation is always in the Topic Map linking to external sites. Ontopia uses a frame-based HTML approach for displaying related links. Another commercial provider of a similar product is *empolis* with their *k42* knowledge server and the *X2X* link server[3]. The empolis k42/X2X architecture is similar to XLinkbase and

[1]See http://www.activenavigation.com/

[2]See http://www.ontopia.net/

[3]See http://www.empolis.com/

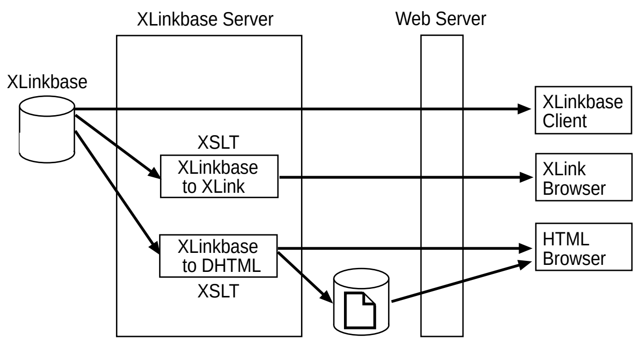

Figure 1: Architecture of the XLinkbase Server

could be used to implement DHTML link visualization, but to our knowledge no such attempts have been made so far.

The link visualization proposed in this paper differs from the first two systems mentioned above, as it is not necessarily relying on any client side applications and still provides the full benefits of multi-ended linking and external linkbases. By using DHTML, it also gives the user the possibility to have a one-click interface to extended links. There is a close integration into the browser, and also into the way Web content is viewed today.

Also, extended links can be included into any page and navigation. It is therefore not limited to the representation of the Topic Map, as implemented in Ontopia.

The proposed system combines the simplicity of the Web with the power of an underlying OHS architecture.

3 An Example for an OHS — XLinkbase

Generally speaking, the interface introduced in this paper can be adapted to display extended links of most OHS systems, especially if they store information semantically connected. However, it has been primarily designed and implemented for the XLinkbase system. The underlying model is therefore shortly explained to illustrate the background of the work.

The overall concept of the XLinkbase system is that of an OHS architecture, with a database of information about resources (which are not part of the database) that can be accessed and presented with a variety of clients. Currently, three major types of clients have been be identified (see Figure 1), which are

- a special XLinkbase client (implemented as a COM component embedded into the Internet Explorer) providing a powerful graphical visualization of XLinkbase data as well as supporting its maintenance and manipulation,
- XLink-capable browsers (unfortunately, not yet available), which could display XML documents with embedded XLinks generated from XLinkbase data,
- and regular HTML browsers, which are supported by transforming XLinkbase data into HTML or DHTML, depending on the requirements. Possible implementations include

– a simple HTML interface that allows the navigation through XLinkbase content using basic HTML linking[4], and
– applications which query the XLinkbase and then generate hypermedia content as required, that can be accessed via any browser providing current state-of-the-art DHTML functionality[5].

Furthermore, the (D)HTML browsers could be supported by either using pre-computed link information, or having it generated on demand (this issue is discussed in more detail in Section 5.1).

It is important to recognize that XLinkbase is specifically designed to support direct navigation through the link data through the first two types of clients, as well as providing the foundation for content management systems for automatically generating rich hypermedia for use with regular (D)HTML browsers.

In the following sections, the XLinkbase system is described in greater detail. A short overview of the system's server-side is provided, followed by describing clients, in particular a Web-based implementation.

3.1 XLinkbase Server

The categorization of meta data about resources has a very long tradition, ranging from early library catalogs over standardized concepts for thesauri [18, 17] to the latest development in this area, W3C's *Resource Description Framework (RDF)* [20, 4] and the ISO/IEC Topic Maps standard [19] as well as its XML variant *XML Topic Maps (XTM)* [28]. While we think that Topic Maps provide a very useful foundation for describing meta data, we also think that there are some areas where Topic Maps are less than ideal. In particular, among the well-known weaknesses of Topic Maps [30], two issues were most important for us:

- *The ability to have a consistent and well-designed hierarchy of topics.*

  Topic Maps support topic types, but do so in an inconsistent way, because (1) a topic is not required to have a type, and (2) there is no built-in mechanism for associating topic types themselves (no defined subtype mechanism).

- *A system for defining constraints for topics and associations.*

  Topic Maps do not make any assumptions about the facets of a topic or the topics being associated by an association, and while this freedom in some scenarios may be useful, it often is too lax and encourages the creation of poorly structured Topic Maps.

Based on these observations, we have created a data model which is similar to Topic Maps, but avoids their disadvantages. Taking this data model as the foundation, we have furthermore specified an operational model, which defines the operations which can be executed on XLinkbase data. The XLinkbase server is an implementation of a system that maintains XLinkbase data and processes XLinkbase operations. In the context of this paper, the most

[4]See http://wildesweb.com/glossary/ for a working example.

[5]When XLink/XPointer-enabled browsers become available, it can easily be adapted to them by generating XLinks rather than DHTML from XLinkbase data.

important property of the server is its ability to apply XSLT transformations to the results of queries into XLinkbase data.

The XLinkbase server is implemented based on *Enterprise JavaBeans (EJB)*, which makes it possible to easily add scalability and distributed operation of XLinkbase servers. XLinkbase JavaBeans communicate by exchanging messages containing XML, and on receipt of a request a special dispatching JavaBean computes a path which a request has to take, and then forwards the request to the first JavaBean on this path. The path may contain many JavaBeans, which all provide a special functionality (access control, logging, notifying external applications). On the back end, the request is transformed into a query language for the actual data storage (SQL in our first prototype), and the XLinkbase data is retrieved from the DBMS. On the way back through the XLinkbase server, more JavaBeans may process the data, the most important being JavaBeans applying XSLT style sheets to transform the XLinkbase data to other representations. We currently provide style sheets for simple HTML, for DHTML as presented in this paper, and for an alternative *Scalable Vector Graphics (SVG)* [12] representation, which currently requires a special SVG plug-in to be installed in the browser.

3.2 XLinkbase Clients

As mentioned above, the most important design goal of XLinkbase is to implement a system that on the one hand provides a powerful and easy-to-use graphical representation through a native client, while on the other hand being able to support more restricted but also more deployed technologies for leveraging XLinkbase data. This model is shown in Figure 1.

We have implemented an XLinkbase client which is implemented as a Windows COM component, and can be easily integrated into Microsoft's Internet Explorer[6]. This interface is limited to Windows platforms only and can also only be used with Internet Explorer, but makes it possible to integrate XLinkbase content with normal Web content and other data being available through COM components.

However, the topic of this paper is a more widely available technology, and this is supported through XLinkbase data being transformed into DHTML. Using this approach, we make little assumptions about the client's functionality, and XLinkbase content can easily be viewed using a standard Web browser on any platform, provided it supports DHTML (which means HTML, CSS, DOM, and JavaScript). In the following section, we explain the motivation of this design.

4 Extended Link Visualization

Extended link functionality is new to most users of the Web. A special challenge therefore lays in making those concepts known to and accessible for users. A new feature is only accepted by users when it is easy to use and builds on already known concepts. It is also well-know that users are very reluctant (or not capable) to install programs on their computers [26]. Even installing a plug-in can be a serious obstacle for inexperienced computer users. Since the Web is used by the general public, new functionality requiring new software components will only

[6]It is displayed in a separate window in the same way as IE's "History" feature. The client also offers a complete interface for controlling the visualization and performing XLinkbase operations.

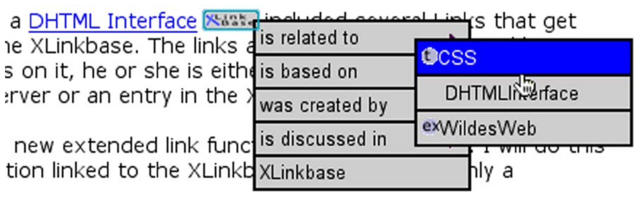

Figure 2: Interface Design of the Prototype

be successful if either the installation mechanism is fool proof, or the software components are delivered pre-configured in the browser.

We therefore reason that for the Web, a Web-based solution, building on existing technology, known metaphors, and accepted user interface widgets, should be provided (while still making more sophisticated interfaces available through native clients).

A functionality similar to multi-noded links already exist in the average-user's computer world: Menus. Here also, users can make several choices under one main point. Depending on the choice they take, they will reach a different destination.

Several different kinds of menus exist [6]. One of particular interest is the context menu used in all kinds of software. The context menu offers actions specific to a chosen part of the application/screen. Depending on the object the user invokes the context menu on, different menus appear next to the object.

Since this is a already known paradigm, it can be used to visualize extended link data. Difficulties are that the menu used for extended links is not actually a context menu offering different actions. There is just one action (linking) and different choices (the different target destinations). Also, we need cascading menus, further stretching the metaphor. However, we believe that this visualization is still intuitively clear to the user. In order to distinguish it from the regular context menus, it is invoked pressing a dedicated icon on the page. This icon visually shows the presence of a more elaborate functionality. Further, we limit the menu to two levels, similar to a regular menu structure, in order to lessen the user's mental burden [6].

The obvious advantages of a context-style menu are that (1) it saves precious screen real estate, (2) it displays the choices for the object adjacent to it (which establishes the mental connection between objects and choices), (3) it does not disturb the layout of the page, (4) it is a known GUI element, and (5) it can easily accommodate the target destinations.

The rationale behind the organization of the information and choices in the menus are described in the following section.

5 Implementation of an Interface

In the following sections, we discuss the technical aspects of the interface, as well as the specifics of our implemented version.

5.1 Technical Discussion

As discussed above, the additional functionality is implemented by relying as little as possible on cutting-edge technology. We therefore developed a DHTML interface only requiring JavaScript on the client side. Also, in order to accommodate users with alternate viewing devices

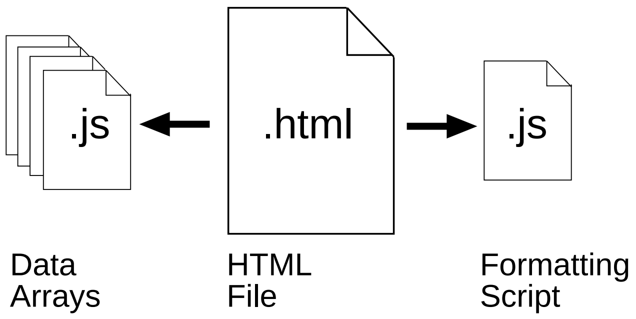

Figure 3: Files for extended Link Functionality

(PDAs, mobile phones, ...) that do not support JavaScript, we have ensured the existence of a way to not only navigate through the information in a traditional way, but also to be able to benefit from the extended functionality of extended linking. This is accomplished by enabling access to an HTML version of the information. To ensure maximum adaptivity to different environments, we have built several modules to be combined easily. The idea is to have three types of files (see Figure 3): One source file with the extended links embedded. This is a standard HTML file which includes several JavaScript Files. For one there are the computational and formatting files that produce the pop-up menus. Also required are data files that contain the actual menu content. Each data file contains an array of links.

There are three main use cases for the interface we will discuss. Basically, the key factors lay in how often the linkbase changes, and how much the owner of the linkbase wants clients to access and use the resources (see also Figure 1):

1. *Linkbase hardly updated, mostly static information*

   In this case, we want to minimize the workload of the server. All the resources are stored in pre-calculated JavaScript files that once downloaded work entirely on the client side. The scripts are downloaded only once, when the HTML file is requested. This way, users are ensured an instant feedback to all their actions on the page, once it has been loaded. Whenever the data changes, the resource arrays are recalculated and stored on the server, so users must reload the page to get the updated data.

2. *Linkbase very frequently updated*

   If the linkbase is changed frequently and it is crucial that the clients always have the latest information (eg, financial market data information), then recalculating all the arrays for all changes is not very economical. Rather, the arrays are calculated on demand, whenever there is a request. To achieve this, a CGI script is executed on the server whenever the page is loaded, updating all arrays relevant to the page to the latest information in the Linkbase before they are downloaded to the client. This of course requires more performance on the server side the heavier the traffic is. Also, additional delay might push the limit on the amount of time the user has to wait. This solution has to return the page within about 10 seconds to be accepted by users [25,32].

3. *Very Large Number of Links on one Page*

   The problem of performance is even more severe if the HTML file includes a large number of links. The overhead is extremely high, as probably only one link will be followed, several may be examined, making the other updated arrays oblivious. A way to cut down on the computational overhead is to load and update the array on demand, that is, when the link is clicked. This lowers the overhead and download time required substantially. However, the time-delays accompanying this form of interaction can be substantial. The user expects an immediate reaction to a click on a menu (as it occurs on other electronic systems), and could be irritated by the lack of immediate feedback on the screen. Therefore, a good feedback mechanism has to be put in place. We recommend this technique only in situations where bandwidth as well as delay are not a problem (eg, fast LANs).

5.2 The Implemented Version

To answer the question on how to present the linking information contained in an linkbase, we have to look at it from a users point of view. Basically, what users expect from the extended link functionality is that they can reach the most appropriate resource from their current location following a link. We therefore need to provide a view of the relevant part of the linkbase. This, however, is not trivial. The problem is that the linkbase server does not know what the users want. So we can either ask the users to state their goals or preferences, as implemented in various systems (for example the COOL link model [23]) or the creator/machine assumes the scope of interest and relevance using for example weighted arcs. In order not to overwhelm the user with choices, the system should not present the user with more than 7 choices per menu or sub-menu [13]. This is to ensure a fast navigation with low cognitive load for the user considering the 7$\pm$2 rule of cognitive psychology [22].

5.2.1 Choosing Data to Display

Generally speaking, the interface can display all sorts of information from different OHS. Depending on the system, different data will be displayed. Key requisite and common denominator of extended links is that the choices are semantically related to the origin.

Using XLinkbase as an example, the selection of the data to be displayed is highly configurable according to the specific demands and needs of the linkbase. We only discuss one possible solution which in our opinion will be feasible in most cases. It is based on several filter mechanisms and criteria. Firstly, we draw a radius of one association around the current topic (see Figure 4). This has the disadvantage of limiting the user's horizon, yet having more than one level of menus normally confuses the user [6]. Simply restricting the radius, however, might not be a good enough filter if a topic is associated with hundreds of other topics. Therefore other means have to be put in place. The creator of the page can use self-defined filters to display a tailored view of the information. This also allows the display of different views according to context. For example, one association type might be the preferred one in a certain context, hence the topics connected to it are privileged. Even an exclusive use of one association is possible. If no filters are defined, the system uses defaults. The creator of topics can also assign weights to the arcs connecting the topics. This information is used in a second stage to further specify the data displayed. If no weights are assigned and/or there are still too many items to display, the user has the possibility to view the HTML representation of the relevant part of the linkbase.

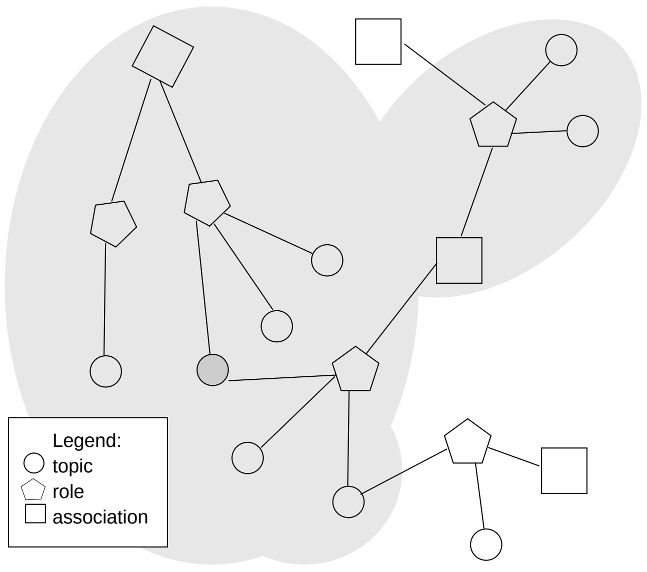

Figure 4: The radius visible to the user

5.2.2 The Link

A standard XLinkbase link in an DHTML interface looks like this:

```
<a href="http://scholastic.com/harrypotter">H. Potter</a>
<a href="/xlbmap.cgi?harrypotter"
   onClick="createMenu('HarryPotter');return false;">
<img src="/xlinkbase.gif" border="0" alt="XLinkbase">
</a>
```

An XLinkbase link consists of two parts. Firstly, the link pointing to the most relevant resource is hard coded into the page on creation or request. This link is a regular HTML link. Users not being able to view the page with the enhanced functionality (alternative access devices, older browsers, JavaScript turned off) will still be able to navigate through the system. Also, this link serves as a trail mark for people not interested in the advanced, more complicated mechanism. The link is determined by the creator or the system comparing weights of arches choosing a "good link".

The second part is a linked image. This visually indicates to the user that there is more and non-standard information available. Users not having JavaScript functionality built in their access device are presented an additional page showing an HTML representation of the relevant part of the linkbase. Is JavaScript available, a menu pops up next to the icon showing the extended link content.

5.2.3 The Menu

The menu is displaying the various target resources to a link. Based on the underlying technology, the menu can either be just a plain list of resources or — in a Topic Map based environment — show associations and the associated resources.

The cascading nature of the menu lets the user traverse via the associations to the resources. The display of the association instead of just a series of links lets the user see the context in which the resource is valid. Therefore, the user has a better information base on which to decide. An example for an XLinkbase menu is depicted in Figure 2.

The resources are normally ordered alphabetically. This is because ordering something alphabetically is one of three precise mechanisms to order information [31]. The system's logic will therefore be intuitively clear to the user. Alternatively, if weights are used in the linkbase, the ordering can also occur according to the weights of each arc. However, this has to be communicated to the users for them to understand the ordering. In XLinkbase, the last item in the association menu points to an HTML representation of the topic in focus. This item is also used when there is too much information available to use the menu GUI.

Since we have two different types of references (internal references to other topics in the XLinkbase and references to external information), we need to visually distinguish the two types. This also to meet the users' expectation of what to find when following a link [6]. We do this by showing an appropriate icon in front of the resource (either 'ex'ternal or 't'opic).

6 Challenges and Future Work

The implemented version as presented here is a big step towards a better linking model for the Web. However, there are still a few limitations and constraints:

- Filtering mechanisms

  If there is a large number of associations for a particular topic, the system has to filter the possibilities to find the most relevant information. This is possible by weighting the arcs (as described in Section 5.2.1), yet requires extra effort on the creator side. Even then, some choices have to be eliminated for the sake of a lean navigation. Filter mechanisms have to be found to ensure the best possible solution.

  Additional comfort and usability of the system could also be added by letting the system decide on a context basis or by evaluating the users' preferences and goals what links to display.

  While filtering has to be performed primarily in the underlying system, it nevertheless is an important issue to consider when thinking about a usable representation of extended links.

- User Testing

  The interface has not been tested systematically by users yet. Since we are introducing a new concept for presenting linking information, the reactions of users to the new functionality would be interesting for the acceptance of our approach in the general public, and valuable data could be extracted to incrementally improve the interface.

While we currently focus on DHTML-based link display, our long-term focus is to also support more sophisticated linking mechanisms, which will become available in the next years [34]. We think that the experiences with our DHTML implementation of complex links will be extremely valuable for creating useful XLinks.

Future XLink/XPointer-enabled browsers probably will support XLinks natively, and while it is not yet clear how they will present XLinks, it is clear that designing an uncluttered and usable interface for the complex linking facilities supported by XLink is a challenge. We hope that our work is a first step towards this goal.

However, when implementing an XLink/XPointer-enabled interface, even more challenges lie ahead, like the issue of properly presenting transcluded content, and the issue of being able to distinguish between internal and external links, which we did not address in this paper.

7 Conclusions

Our work shows that the easy-to-understand and easy-to-use visualization of data from a complex hypermedia system can be challenging. However, we believe that the linkbase-assisted generation of more complex links than today's HTML links is the future, and while we provide a tool-set for the smooth transition to this new generation of links, the ultimate goal is to generate XLink from XLinkbase data instead of DHTML. However, it will take a least a couple of years until XLink/XPointer-enabled browsers will dominate the market, and until this time, there still is a need for visualizing extended links in using today's browser technology.

Acknowledgements

We would like to thank the XLinkbase project team, in particular Yves Langisch and Manfred Meyer for working on the server implementation, and Simon Künzli and Peter Zberg for implementing the native client as well as working on the SVG presentation.

For the visual representation of the links in our first prototype, we adapted Peter Belesis' *Hiermenus* JavaScript Code[7].

References

[1] Kenneth M. Anderson. Integrating Open Hypermedia Systems with the World Wide Web. In *Proceedings of the 1997 ACM Conference on Hypertext*, pages 157–166, Southampton, UK, April 1997. ACM Press.

[2] Michael Bieber, Fabio Vitali, Helen Ashman, V. Balasubramanian, and Harri Oinas-Kukkonen. Fourth Generation Hypermedia: Some Missing Links for the World Wide Web. *International Journal on Human Computer Studies*, 47(1):31–65, July 1997.

[3] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, and Eve Maler. Extensible Markup Language (XML) 1.0 (Second Edition). World Wide Web Consortium, REC-xml-20001006, October 2000.

[4] Dan Brickley and R. V. Guha. Resource Description Framework (RDF) Schema Specification. World Wide Web Consortium, Candidate Recommendation CR-rdf-schema-20000327, March 2000.

[5] Leslie A. Carr, David C. De Roure, Hugh C. Davis, and Wendy Hall. Implementing an Open Link Service for the World Wide Web. *World Wide Web Journal*, 1(2):61–71, 1998.

[6] Alan Cooper. *About Face: The Essentials of User Interface Design.* IDG, Boston, Massachusetts, August 1995.

[7] Hugh C. Davis. To Embed or Not to Embed. *Communications of the ACM*, 38(8):108–109, August 1995.

[7]Freely available at http://www.webreference.com/

[8] Hugh C. Davis, Wendy Hall, Ian Heath, Gary J. Hill, and Robert J. Wilkins. Towards an Integrated Information Environment with Open Hypermedia Systems. In *Proceedings of the Fourth ACM Conference on Hypertext*, pages 181–190, Milano, Italy, November 1992. ACM Press.

[9] Steven J. DeRose, Eve Maler, and Ron Daniel. XML Pointer Language (XPointer) Version 1.0. World Wide Web Consortium, Candidate Recommendation CR-xptr-20010911, September 2001.

[10] Steven J. DeRose, Eve Maler, and David Orchard. XML Linking Language (XLink) Version 1.0. World Wide Web Consortium, Recommendation REC-xlink-20010627, June 2001.

[11] Doug Engelbart. The Augmented Knowledge Workshop. In Adele Goldberg, editor, *History of Personal Workstations*, pages 187–236. ACM Press, New York, August 1988.

[12] Jon Ferraiolo. Scalable Vector Graphics (SVG) 1.0 Specification. World Wide Web Consortium, Recommendation REC-SVG-20010904, September 2001.

[13] Wilbert O. Galitz. *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques.* John Wiley & Sons, Chichester, England, December 1996.

[14] Kaj Grønbæk, Lennert Sloth, and Niels Olof Bouvin. Open Hypermedia as User Controlled Meta Data for the Web. In *Proceedings of the Nineth International World Wide Web Conference*, pages 553–566, Amsterdam, Netherlands, May 2000. Elsevier.

[15] Kaj Grønbæk, Lennert Sloth, and Peter Ørbæk. Webvise: Browser and Proxy Support for Open Hypermedia Structuring Mechanisms on the World Wide Web. In *Proceedings of the Eighth International World Wide Web Conference*, pages 253–267, Toronto, Canada, May 1999. Elsevier.

[16] Wendy Hall, Hugh C. Davis, and Gerard Hutchings. *Rethinking Hypermedia: The Microcosm Approach.* Kluwer Academic Publishers, Boston, Massachusetts, May 1996.

[17] International Organization for Standardization. Documentation — Guidelines for the Establishment and Development of Multilingual Thesauri. ISO 5964, 1985.

[18] International Organization for Standardization. Documentation — Guidelines for the Establishment and Development of Monolingual Thesauri. ISO 2788, 1986.

[19] International Organization for Standardization. Information technology — SGML Applications — Topic Maps. ISO/IEC 13250, 2000.

[20] Ora Lassila and Ralph R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. World Wide Web Consortium, Recommendation REC-rdf-syntax-19990222, February 1999.

[21] Hermann Maurer. *HyperWave — The Next Generation Web Solution.* Addison-Wesley, Reading, Massachusetts, 1996.

[22] George A. Miller. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *The Psychological Review*, 63(2):81–97, March 1956.

[23] Michael Miller and L. Jay Wantz. Computed Web Links: The COOL Link Model. In Peter Brusilovsky, Alfred Kobsa, and Julita Vassileva, editors, *Adaptive Hypertext and Hypermedia.* Kluwer Academic Publishers, Pittsburgh, Pennsylvania, June 1998.

[24] Theodor Holm Nelson. *Literary Machines.* Mindful Press, Sausalito, California, 1982.

[25] Jakob Nielsen. *Usability Engineering.* Morgan Kaufmann Publishers, San Francisco, California, October 1994.

[26] Jakob Nielsen. *Designing Web Usability: The Practice of Simplicity.* New Riders, Indianapolis, Indiana, November 1999.

[27] Peter J. Nürnberg and John J. Leggett. A Vision for Open Hypermedia Systems. *Journal of Digital Information*, 1(2), 1997.

[28] Steve Pepper and Graham Moore. XML Topic Maps (XTM) 1.0. TopicMaps.Org Specification xtm1-20010806, August 2001.

[29] Dave Raggett, Arnaud Le Hors, and Ian Jacobs. HTML 4.01 Specification. World Wide Web Consortium, Recommendation REC-html401-19991224, December 1999.

[30] Hans Holger Rath. Topic Maps: Templates, Topology, and Type Hierarchies. *Markup Languages: Theory & Practice*, 2(1):45–64, 2000.

[31] Louis Rosenfeld and Peter Morville. *Information Architecture for the World Wide Web.* O'Reilly & Associates, Sebastopol, California, March 1998.

[32] Maureen C. Stone, Ken Fishkin, and Eric A. Bier. The Movable Filter as a User Interface Tool. In *CHI '94: Proceedings of the ACM Conference on Human Factors and Computing Systems*, pages 306–312, Boston, Massachusetts, April 1994. ACM Press.

[33] Norman Walsh. XML Linking and Style. World Wide Web Consortium, Note NOTE-xml-link-style-20010605, June 2001.

[34] Erik Wilde and David Lowe. *XML, XLink, and XPointer: A Practical Guide to Web Hyperlinking and Transclusion.* Addison-Wesley, Reading, Massachusetts, to be published in 2002.