Augmenting the Web through Open Hypermedia

Niels Olof Bouvin
Department of Computer Science, University of Aarhus,
Åbogade 34, DK8200 Århus N, Denmark
email: n.o.bouvin@daimi.au.dk; Fax (+45) 8942 5624

Abstract

Based on an overview of Web augmentation and detailing the three basic approaches to extend the hypermedia functionality of the Web, the author presents a general open hypermedia framework (the Arakne framework) to augment the Web. The aim is to provide users with the ability to link, annotate, and otherwise structure Web pages, as they see fit. The paper further discusses the possibilities of Web augmentation through the description of a number of experiments performed with an implementation of the framework, the Arakne Environment.

1 Introduction

The success of the Web can, in part, be attributed to its simple architecture: A stateless file transfer protocol (HTTP), an universal naming scheme (URL) and an easily understood document format (HTML). However, the very simplicity of HTML links (inlined, unidirectional, and untyped) is a liability as the creation of links depends on write-access to documents. In situations, where users do not have write access to documents, or where disjunct sets of links are needed on the same body of documents, the simple HTML link does not suffice. There may be a lot of information available on the Web, but users are not free to annotate these pages as they may with a book. Consider Web journalism and Web logs, that largely consist of references (with short introductions) to pages on other Web sites. In these cases, the authors are limited when referring to other Web pages: they can either make whole page references, or copy and paste relevant parts into their own documents, leaving it as an exercise to the reader to establish the relationship between the quoted portion and the original Web page.

Many systems have been built to address some of these issues, ranging from gathering information about Web pages, storing bookmarks or snippets, organising a large number of URLs, annotating Web pages, creating guided tours, creating bidirectional multi-headed links etc. These activities may be termed "Web augmentation" (1) as they seek to extend or augment the functionality inherent in the Web. Some of these tools have become so commonplace that Web browsers without them would be considered unusable (e.g., organising bookmarks hierarchically, or navigating a browsing session using back and forward buttons), whereas others remain exotic (e.g., adding your own links and comments to other people's Web pages). For a general discussion of this subject, see (1).

The very success of the Web has raised the question whether there is still room for (open) hypermedia research (2). This paper argues that this is the very best of times to be doing research in open hypermedia, as the Web provides researchers with much that was missing before. As a subject for open hypermedia research, the Web has very nice properties. Most of the involved file formats and protocols are open (a relief for a community used to dealing with proprietary editors and formats), the number of Web browsers in widespread use is quite limited, and there is a decent global naming scheme for documents and resources. Given the size of the content available (much of it fairly poorly structured (3)), this is the time where the fruits of open hypermedia may be harvested. Hypermedia becomes really interesting and useful with large collections (4), because this is where good structuring tools really pay off. Not only is there a lot of stuff out there, there is also a lot of people using it (according to some estimates $\sim$ 600 million). Organisations and companies are actively moving their documents on internal and external Web sites. The Web has provided open hypermedia researchers with what we could scarcely dream of before—a unified fairly well-behaved document space with many users.

This paper is structured as follows: Section 2 describes the work done by the open hypermedia community with special regards to the Web; Section 3 introduces the notion of Web augmentation and outlines the general techniques and uses; Section 4 introduces the Arakne Environment, a general framework for developing Web augmentation and describes some of the experiments done with it; Section 5 considers the future of Web augmentation; and the paper is concluded by Section 6.

2 Open Hypermedia and Web Augmentation

The hypermedia extension of existing systems is the field of open hypermedia. Historically, hypermedia systems have often been closed and monolithic, requiring other programs to comply to the standards of the hypermedia system in order

to provide hypermedia functionality. This is problematic as it closes the door on existing non-compliant applications, and expects developers to change their programs—an unlikely proposition at best. Recognising that most people are not willing to throw away their applications in order to utilise a hypermedia system (5) led the hypermedia community to integrate existing applications into their (now open) hypermedia systems instead. The level of which this is possible varies accordingly to the application, ranging from the simple (show document) to the advanced (a full integration), as described by Whitehead in (6). Well-known open hypermedia systems are Microcosm (7), Devise Hypermedia (8), HyperDisco (9), HOSS (10), and Chimera (11).

A natural consequence of the open hypermedia approach is the interest of integrating the Web with open hypermedia systems. Several groups in the field have integrated the Web, and a non-exhaustive list includes DLS and descendants (12–14), DHM/WWW (15), Webvise (16), Chimera (17), and the Arakne Environment (18).

A common approach of these systems is to modify Web pages while en route to the Web browser. This modification usually consists of the addition of links or other kinds of structure, stored on a structure server, and inserted into the pages by means described in Section 3.2. The interface presented to the user varies, ranging from no interface at all to full authoring applications allowing the users to modify and extend upon the existing collections of links, annotations, and other structures. This is Web augmentation.

3 Web Augmentation

Web augmentation is the central theme of this paper, and this section introduces the term and argues for its relevance on the current Web.

Web augmentation tools allows users to create and use external hypermedia structures imposed on Web pages (i.e., not previously present on the pages), that they themselves not necessarily have write-access to. This empowers users as they become able to use the Web in a more flexible way. Given Web augmentation tools, it becomes possible to annotate, link, and otherwise structure all Web pages. Tools could also allow users to share their links with others, or professionals to publish their work to their colleagues and clients, or the general public.

3.1 The Need for Web Augmentation

Before contemplating Web augmentation it is necessary to establish that it is worth the effort i.e., that provides users with useful functionality. This point is illustrated by two scenarios:

> Many organisations deploy intranet with Web servers accessible only from the company network. If several groups within a project depend on the same technical specification, a Web server is well suited for making these documents easily accessible. However, the Web is not only documents, it is also links. Consider therefore the situation where several of the groups wish to insert links into the technical specification. This can be accomplished in several ways using standard Web technologies. Either all links are placed in the same document, or identical copies of the specification are made for each group. Both solutions are unsatisfactory. In the former case, all groups will either have write-access to the specifications in order to add links, or possibly send their links to a single Web master. Both solutions have both overhead and the possibility of accidently corrupting either the specification or other groups' links. In the latter case, each group has their own copy. This too is unsatisfactory, if the specification at a later date changes, as n copies of the specification then must be updated. It could also be that the specification is so important, that it may not be modified at all to prevent accidental changes.

This scenario illustrates that while Web technology is supremely well suited for *distribution* of material and documents, it is not well suited for (collaborative) *work* with these documents. Given a tool for creating links (and other structures) upon Web pages, the groups described above would have no trouble sharing a single set of specifications as long as they had separate sets of hypermedia structures.

> John Hansen is an agricultural consultant, and he spends most of his time researching and writing reports and articles for the farming community. His company has in the last couple of years started to focus on the Web and John is currently working on the Web version of his most recent article about herbicides. John is not quite satisfied with the contents of the Web pages from one large herbicide manufacturer, and decides to put in a comment and a link on the manufacturer's Web page to his own analysis. Later, when John publishes his article, users of his company's site (and their proxy) can see his comment and link on the manufacturer's Web site. While he is at it, John creates a guided tour juxtaposing some recent video clips from national TV of unsubstantiated claims with regards to the use of the herbicide "Ground Up" with the actual test reports and results, while adding his own comments.

This scenario illustrates how Web augmentation could be used for critical reading and for structuring the Web for others. John plays the role of Bush's

"Trail Blazer" (19), as he creates his annotations, links, and guided tours for his readers. For another take on the need for this kind of functionality, see (20).

Two issues occasionally raised in the context of Web augmentation are intellectual property and copyright. Is it legal to modify other people's Web pages? It is here important to remember that an essential part of Web augmentation is that no modification to the *actual* Web page is taking place—it is not technical possible (barring write access) and it would defeat the purpose and benefit of externally stored structures. It can therefore not be compared to e.g., Web site defacing perpetrated by hackers. On the other hand, the ability to distinguish between primary (the Web page itself) and secondary (external links, annotations etc.) material is important, and it is the responsibility of both the tool developer and the annotator to support this (see Section 4.4). Given that most Web augmentation tools require special installation or configuration, users in general would not be in doubt that they are using such tools and that a Web pages therefore may not be quite what it seems. However, if the existence of the tool is transparent or if the user has no influence on what material is presented, the situation is different. This was illustrated by the Microsoft Smart Tags case (21)—where the announced generic link functionality was disabled due to an outcry over the perceived loss of control.

3.2 Methods of Augmentation

Assuming that Web augmentation may be desirable, let us then consider the methods that may be employed to provide users with this extra structuring functionality.

In general, Web augmentation tool developers wish their tool be remain active and relevant to what the user is doing with the Web. Thus, the system must be kept aware of what Web page is being displayed, and may also need some way of modifying the contents of the current page. These seemingly simple requirements have not been without some difficulties during the progression of the Web and the varying dominant Web browsers.

The architecture of the Web is well-known, and the methods of augmentation follow directly from this architecture. This section gives an overview of the three basic approaches, compares their advantages and disadvantages, and describes systems that employ these methods.

3.2.1 Server side

Web content originates from a Web server, and given access to the server, it is simple to modify this content dynamically, e.g., through the use of server side scripting. As the functionality resides on the server, the content provider can easily modify and extend the available services. A classic example is the guided tour system Walden's Paths (22). The users are guided along the trail through a

navigation bar inserted at the top of all Web pages by the server. An example (unrelated to hypermedia research) of providing the user with customised information is the Amazon.com recommendation system, which, based on the user's previous purchases, suggests other products likely to appeal.

The main advantages of the server-side approach are simplicity, accessibility, and control. There is no need for any special Web browser configuration and the pages can (in principle) be accessed from any browser. The functionality resides completely on the Web server, so the content provider can fully control the availability of the services.

Among the disadvantages are general applicability, robustness, and authoring interface. If one wishes to augment Web pages from more than one Web server, these "foreign" pages must be modified in some way. One typical approach (taken by Walden's Paths) is to let a server script retrieve the Web page and modify all links encountered to invocations of server scripts with the original link as an argument. However, the use of e.g., frames or bookmarks in the Web browser defeats this modification of links (as the page retrieval would bypass the server), and thus all actions by the user cannot be handled through the server-side approach. Similarly, the authoring of hypermedia structures becomes difficult, as there is no interface save the Web page itself. The common approach to this is either to use a separate authoring tool (this is the case of Walden's Paths), or to provide special Web pages with forms allowing the authoring of e.g., links, though the latter approach is cumbersome. Still, if the purpose is to augment a specific Web site and authoring is to be handled by a few, server-side Web augmentation is effective.

3.2.2 Proxy side

Web proxies are widely used for caching purposes, but can also be used to modify Web pages. The great advantage is accessibility and availability—once the Web browser has been configured, the system is always on. This is also one of the greatest drawbacks—it is quite plausible that the user does not wish to constantly use the proxy, as the querying of link bases inevitable will affect performance.

The prototypical example of this approach is the Microcosm Distributed Link Service (DLS) system (12). In this case the proxy acts as a hypermedia client, retrieving links etc. from link bases and inserting them into Web pages. Another hypermedia proxy is DHMProxy (23). An interesting feature of DHMProxy was the method of link decoration (i.e., the act of inserting links etc. into the Web page). Based on information retrieved from the hypermedia server, the proxy inserted JavaScript code into the Web page, which upon arrival was executed and performed the link decoration. This was done to circumvent the problem described in Section 3.2.3 with locating anchors on dynamic Web pages.

A flexible approach to using proxies as "intermediaries" is described in (24),

where a chain of proxies can be used for arbitrary computations on the retrieved Web pages, ranging from more meaningful error messages (rather than e.g., "Error 404"), over inserting annotations, to collecting statistics about users' browsing habits. The intermediaries may be local to the user's machine or shared among between the members of a workgroup or organisation.

The reservations with regards to hypermedia authoring noted above also hold for proxy solutions, e.g., DLS used a separate Web page for link authoring. One example of a hybrid solution was an early version of the Arakne Environment (see Section 4), which used DHMProxy for link decoration. In order to use a proxy solution, the user has to setup the Web browser to use that particular proxy. This has some problems: The user may not want (or may not be able) to make this configuration change; the use of a special proxy may collide with the use of another proxy; and the user will feel the performance hit of the hypermedia proxy, whether the current Web page has any extra hypermedia content or not. The performance hit of the proxy becomes especially apparent when retrieving large multimedia files, as these also must pass through the proxy.

3.2.3 Client side

Web pages can also be modified at the client. The first system to do so was DHM/WWW (15), and the client-side approach has also been taken by its descendants, Navette (25), and the Arakne Environment (1), as well as Webvise (23). Other systems to use this approach include the commercial annotation tools iMarkup and Third Voice, both integrated with the Microsoft Internet Explorer.

There are two approaches to client side Web page modification: Pre- and post-render, that is before and after the Web browser has displayed the page. DHM/WWW (15) and Navette (25) utilised a pre-render approach, DHM/WWW by retrieving the page, modifying it, and generating JavaScript to output it to the Web browser; and Navette by acting as a local proxy. Both instances can thus be argued to be quasi-proxy solutions, both with the performance hit of having to compute all links, etc. before rendering the Web page. In contrast, Webvise and the Arakne Environment use post-rendering page modification. There are several advantages to this approach. The first advantage is that of performance, as pages are displayed in the Web browser as fast as they would without the hypermedia application. The second advantage is the ability to more robustly handle "difficult" HTML. Many Web pages contain JavaScript, which upon arrival is executed and modifies the page. This makes it difficult to reliably locate anchors pre-render, but simple after the Web browser has rendered the page.

Another advantage of a client side solution is that authoring becomes very straightforward to support. Client side necessitates a hypermedia application running along or within the Web browser, and this application can of course also

accommodate authoring with as rich an interface as desired.

The two greatest disadvantages of client side are the need to download and install special software, something users may not be willing (or allowed) to do; and the specialisation to one Web browser (as there is no usable standard API for close Web browser integration). Given the current dominance of the Microsoft Internet Explorer, the latter may seem as less of a problem, but it remains a concern for generality and future development, especially with the advent of Web browsers in mobile phones and other devices.

3.3 Summary

The three basic methods of Web augmentation outlined above are all valid and usable in different domains—there is no one right way to do it. By relying on a server side solution, a content provider can provide users with added value "out of the box". Proxies have their strength in high availability, once they have been configured. While they may not scale up to massive use, they are well suited to provide e.g., a department with shared hypermedia functionality. For maximal control, there is no alternative to a full integration with a Web browser, which opens up for a rich set of possible manipulations upon Web pages, plus supporting a sophisticated authoring interface. Often the most practical solution will be composite—e.g., using a client side application for authoring, and a proxy for publication.

4 The Arakne Environment

The Arakne Environment, seen in Figure 2, has been described in (1, 18, 26, 27). It is a collaborative component-based open hypermedia system written in Java aimed at augmenting the Web with externally stored hypermedia structures, such as bi-directional multi-headed links, guided tours, or spatial hypermedia. The different hypermedia structures are handled by an open set of specialised hypermedia tools, known as "views". The current Arakne Environment relies on the Open Hypermedia Protocol compliant Construct servers, a later version of which is described in (28).

The Arakne Environment has served as platform for a number of Web augmentation experiments described below.

4.1 A General Web Augmentation System

The Arakne Framework can be seen in Figure 1. It is based on the analysis of many, quite different, Web augmentation tools, and models such tools by identify-

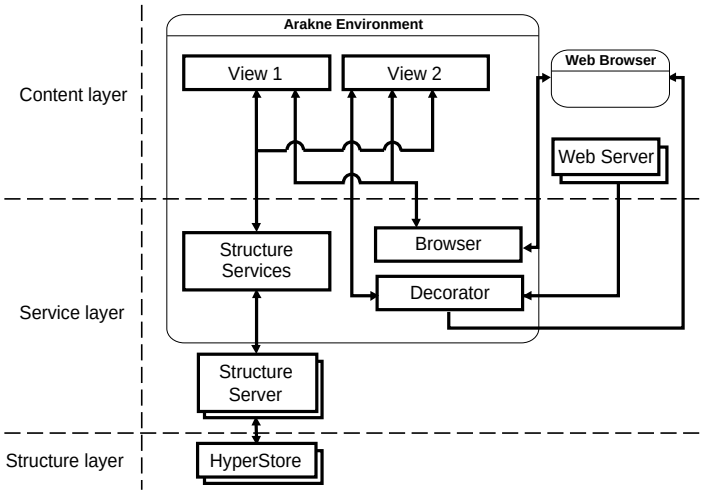

Figure 1: The Arakne Framework. The framework models the task of a Web augmentation tool: providing a user with an interface, keeping track of what the Web browser is displaying, retrieving relevant information and modifying Web pages if necessary. The figure illustrates a situation with two hypermedia views, one of which modifies the pages displayed by the Web browser.

ing the necessary constituent components and their interactions. A Web augmentation tool needs an interface (a view) so the user can interact with it; it must know which Web page is being displayed (handled through the browser component—a wrapper for the actual Web browser); it needs to retrieve the information pertinent to the Web page in question (structure services); and if such information exists, it may need to modify the Web page (decorator). These tasks are reflected in the Arakne Framework. The Arakne Framework can be used to model arbitrary Web augmentation tools, and some examples of this are given in (1).

The Arakne Environment, presented for the first time in (1), follows the Arakne Framework closely in its implementation. The aim of the Arakne Environment has been to create a component-based architecture to support the implementation of Web augmentation tools, allowing prospective developers to concentrate on the more interesting task of developing their tool's core functionality, rather than reinventing the supporting components. As such, the Arakne Environment supports an open set of Web augmentation tools. Hitherto, the system have featured the Navette navigational view (25), the Ariadne guided tour view (29), and the CAOS spatial view (30).

The system has been rewritten on a number of occasions as detailed in (27, 31), extending and adapting the overall system to new requirements. Though the

current Arakne Environment is functionally quite different from the original (the original supported neither collaboration nor the Open Hypermedia Protocol), the overall architecture, based on the Arakne Framework, remains the same. The progression of the general architecture can seen in Figures 1, 5, and 6.

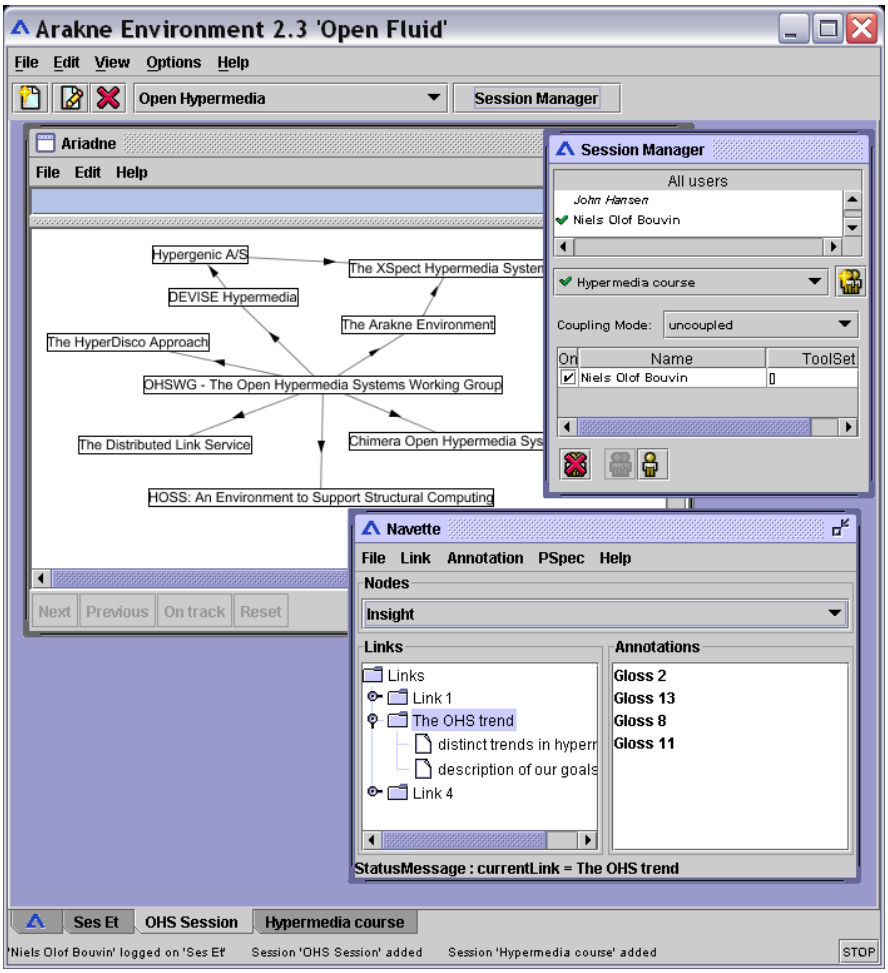

Figure 2: The Arakne Environment: Three sessions, two views, and a Session Manager. The ticker tape is visible at the bottom. The tabbed panes just above the ticker tape are used to switch between joined sessions.

4.2 Collaboration Support in the Arakne Environment

A basic form of collaboration in large document spaces is collaborative structuring and annotation. By structuring a document space, we begin to make sense out of

it. While personal annotations certainly are beneficial to the original annotator, they may also be valuable to others, even when these annotations were not written with the purpose of publication, as described by Marshall in (32). By hypermedia structuring, we impose order, we assert relationships where none were before, we admit a text into a larger context. These are basic characteristics of hypermedia, and characteristics that can only be fully realised in a system where users are free to link, annotate, and structure as they see fit, and where such structures can be shared with others. To support this kind of activity, a number of collaborative hypermedia systems have been created, such as Sepia (33) and HyperDisco (34).

The Arakne Environment relies on the Construct servers (28), which support collaboration. The Construct architecture is based on the work done on the Open Hypermedia Protocol by the Open Hypermedia Systems Working Group (35), and features the data model seen in Figure 3.

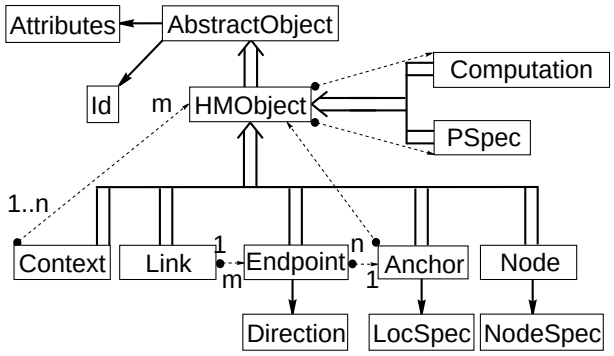

Figure 3: The Open Hypermedia Protocol navigational (OHP-Nav) data model (35) in the Whitehead (36) notation.

The current architecture allows users to be engaged in an arbitrary number of sessions using any number of hypermedia views. Each session has its own data model, coupling mode, and views, independently of other sessions. The coupling mode defines the visibility of each users' actions. The current version of the Arakne Environment supports three coupling modes: Uncoupled, loosely, and tightly, as in Sepia. To supplement the coupling modes, OHP also supports subscriptions on OHP events. This allows a user to e.g., be alerted, when John creates a link, but not when Jim does.

To accommodate this new functionality without disrupting the existing user interface too much, an attempt was taken to make it possible to collaborate without having to focus on the collaborative parts of the interface, such as the Session Manager (seen in Figure 2). To provide users with shared or peripheral awareness of other users online, a tickertape located at the bottom of the window displays

items such as the subscribed events described above, as well as monitors users logging in or out of the system.

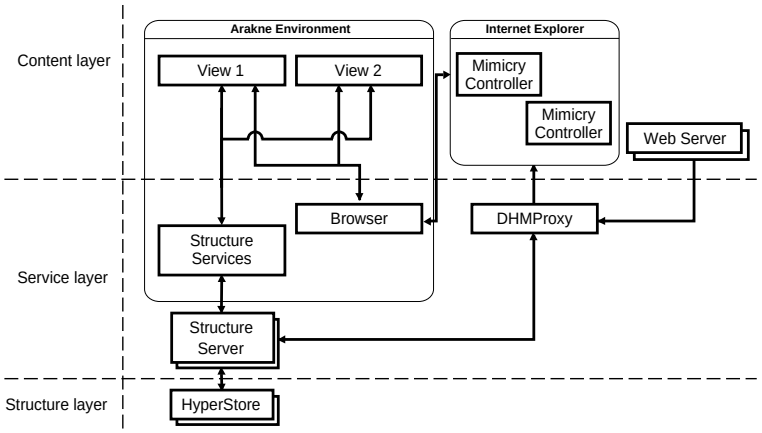

Figure 4: The Arakne Framework with support for media links. The DHMProxy handles link decoration and inserts the Mimicry applet to handle temporal media.

4.3 Temporal Multimedia Links

The Web is more than HTML pages. As the common Internet connection becomes faster, other media types, such as audio or video, become more widespread. In order to demonstrate that the open hypermedia Web augmentation approach could be extended to more than HTML, the Arakne Environment was extended to integrate video files on Web pages as described in (26).

At the time of development, we were faced with the recurring problem of open hypermedia: Content handlers that could not be sufficiently integrated for our purposes. We could do "launch only" integrations, but this was insufficient given the desire to link into and from segments of video files. This necessitated the creation of a special purpose media player applet, which was to mimic the ordinary media player, hence the name Mimicry.

At the point of creation, the Arakne Environment did not do its own link decoration. This were handled by the DHMProxy (23), which in this case identified embedded media clips (identified by `<embed>` or `<object>` tags) of the types handled by Mimicry, and rewrote these to `<applet>` tags, invoking the Mimicry applet to play the media file, as well as specifying the anchors, if any, in the media clip. Thus, the normal plug-ins were circumvented, and a media player with a much richer programming interface was provided. Through integration with the

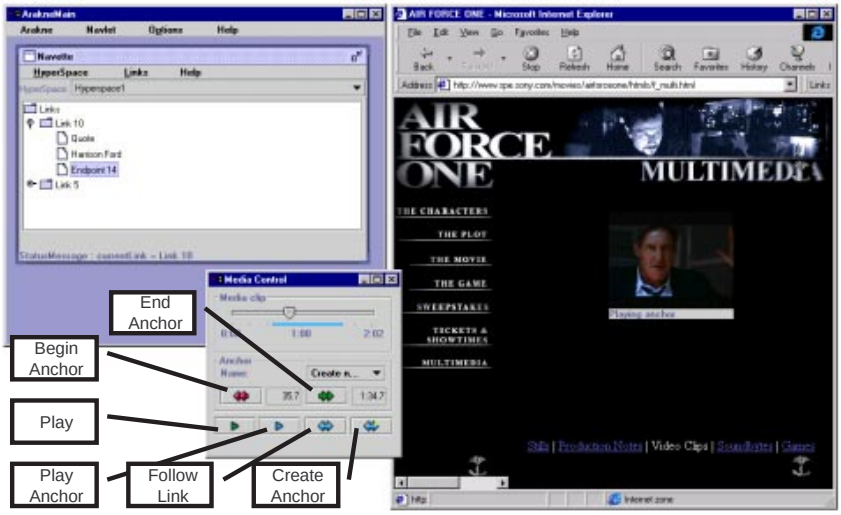

Figure 5: The Arakne Framework with the Mimicry applet. The applet provides an interface for anchors on the time line.

Arakne Environment, it was then possible for users to author links with anchors in ordinary HTML pages as well as in media clips, as seen in Figure 5.

The general architecture of the Arakne Environment at this time can be seen in Figure 4. The main development compared to the Arakne Framework (Figure 1) is the DHMProxy which resides apart from the Arakne Environment.

Since the publication of this work, media players have evolved to a stage, where they provide all the basic functionality for anchoring within media clips. An example of a current much more ambitious system with hypermedia integration of temporal media is the Temporal Linking Service (37), which integrates streaming media with continuous metadata.

4.4 Fluid Annotations for the Web

The support for annotations is widespread among Web augmentation tools. The Arakne Environment has supported annotations since the integration with the Microsoft Internet Explorer. Annotations could be pre- or post-fixed to the anchor, replace the anchor text with the annotation, or provide a pop-up text in the form of a tooltip. This approach had a number of disadvantages. The insertion of the annotation directly into the Web page is disruptive to the original layout of the page, especially as annotations were shown in a single colour and style, chosen by the developers to be distinct from the original page. Tooltip annotations are not

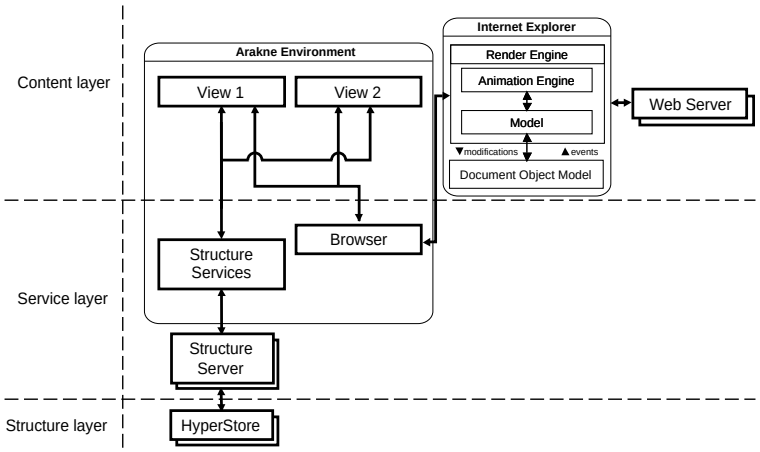

Figure 6: The Arakne Environment with the Render Engine. The Render Engine handles the ordinary link decoration as well as gloss animation through manipulation of the Document Object Model.

intrusive, but they are elusive (i.e., they disappear upon pointer movement), and only one is visible at a time.

A more sophisticated approach is that of fluid annotations (38, 39). Fluid annotations are animated, so that the gloss (i.e., the text of the annotation) smoothly appears and disappears as desired without obscuring the text on the page. Room for the gloss is either found where available on the page (by e.g., placing the gloss in the margin of the page), or room is made available by various means (e.g., compressing the line spacing of nearby paragraphs, or pushing text below further down). When the gloss is closed, the previous state of the page is restored.

From a usability point of view, fluid annotations have several advantages: They do not affect the page when not in use; they can be left open without obscuring other content; and users can copy/paste from opened glosses. See (40) for a usability study of fluid annotations. Earlier versions of fluid annotation prototypes had relied on custom built systems with special purpose document formats, and the Arakne Environment formed a test bed to determine whether the fluid functionality could be realised on the Web. This work is described in (18, 27).

Fluid annotations are by their nature animated, and the link decoration functionality of the Arakne Environment was not able to support such behaviour. In order to handle smooth animations, the link decoration architecture was changed to include a special purpose component, the Render Engine, which was integrated with the Internet Explorer, as illustrated by Figure 6. The Render Engine is a DLL, which communicates with the Arakne Environment through the Java Native

Interface. In addition to the animation of glosses, the Render Engine forms a general interface to Internet Explorer both with regards to ordinary browser events and commands (e.g., display a URL) as well as link decoration and user interface for authoring. Thus, the Render Engine is in essence the Browser and Decorator components from the original Arakne framework rolled into one.

The Render Engine's tight integration with the Internet Explorer also streamlines the authoring interface. Most authoring and browsing of fluid annotations and open hypermedia links can be accomplished directly in the Internet Explorer. The creation of links and glosses are usually handled through context-dependent popup menus. Right-clicking on a text selection and selecting "Create Gloss" in the context menu creates an open hypermedia anchor for the selection and opens an auxiliary window in which the annotator can type the text of the gloss (using HTML, if desired) and select the styling of the annotation. Once the window is closed, the Web page is refreshed and the new annotation appears. Inserting open hypermedia links and adding endpoints to existing open hypermedia links is handled similarly. Endpoints and glosses can also be removed through the context menu. The commands available in the context menu depend on the state of the system (is the user currently authoring a link?) and what element the user has right-clicked on (a new selection vs. an existing endpoint?).

The animation supported by the Render Engine is a pushdown animation where the space between the line containing the anchor and the line below is extended to make room for the gloss. See (18) for an in-depth explanation of how this is accomplished through the manipulation of the Document Object Model.

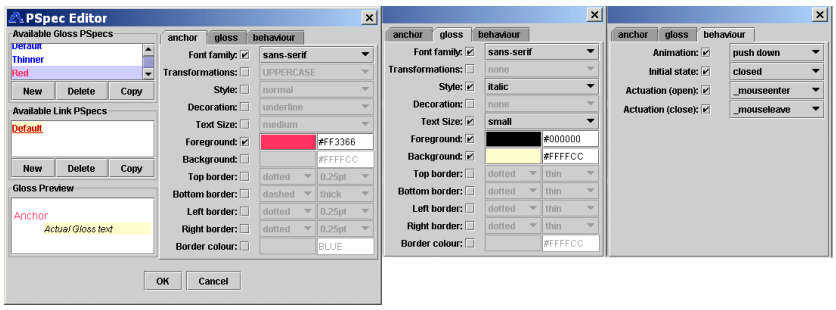

Figure 7: An unfolded view of the Presentation Specification Editor.

The links and annotations added by the Arakne Environment are not ordinary links, and in order to lessen user confusion, they should not appear as such. In order to support a richer set of appearances, the Arakne Environment provides a Presentation Specification (PSpec) Editor, which can be seen in Figure 7. The author of a link or an annotation creates a PSpec (which in this context can be thought of as a Microsoft Word style), and applies it to the link or annotation in

question. As PSpecs are first class objects in the Open Hypermedia Protocol (see Figure 3), a good PSpec can be reused, allowing for a uniform styling of specific classes of links and annotations. In addition to visual appearance, the annotator can also specify how links should be followed and glosses should be opened. The default is "click" for links, and "shift + MouseEnter" for glosses, but it can be any combination of mouse events and modifier keys that the annotator wishes. See (41) for a in-depth discussion of link presentation.

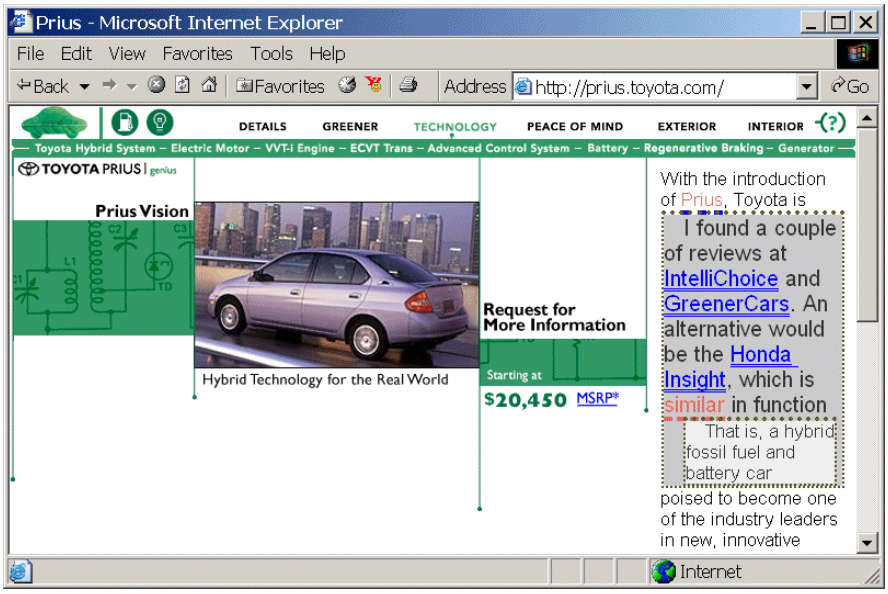

Figure 8: Fluid annotations in use. This screen shot illustrates a situation where a user is researching low emission vehicles. The box on the right is a gloss containing links (double underlined) and one opened gloss.

Annotations do not stand alone, and in a collaborative setting it is natural to comment on others' comments. To support this, the Arakne Environment provides glosses within glosses, i.e., annotated annotations, as seen in Figure 8. This is supported by the generality of the OHP-Nav data model, as an Anchor may have any HMObject as its parent object. The Arakne Environment models gloss text as an attribute of an Anchor. Thus an annotation to an annotation is an Anchor (with a gloss) which has an Anchor (containing the original gloss) as a parent. This allows the Arakne Environment to handle any number of glosses within a gloss.

This work has shown that it is possible to support fluid annotations on the Web. Tests have shown that the animations are fluid even on elderly machines, and that the Render Engine is fairly robust on most Web page layouts. Open fluid annotations showcase some of the possibilities with modern Web browsers, and

how these possibilities can be used for structuring and annotation support.

4.5 Summary

The Arakne Environment has been used predominantly as a test-bed platform for hypermedia experiments. It serves as an example of what is possible with a client side Web augmentation tool, and it showcases some of the advantages with regards to presentation and interface. As an ongoing effort the goal of the Arakne Environment is both to conduct experiments to see what is possible in Web augmentation, and to provide prospective developers with an general architecture with reusable components and services.

5 Future Directions for Web Augmentation

This section describes some of the possible future directions for Web augmentation and externally stored hypermedia on the Web in general.

The future of Web augmentation is bright. The proliferation of standards such as the Document Object Model, Cascading Style Sheets, XML, RDF, XSLT, XPointer, and XLink will only make it easier for developers to continue the work of extending the functionality of the Web. One of the major stumbling blocks for a hopeful Web augmentation developer is the uneven support for these standards in popular Web browsers. Sometimes this forces developers to create such workarounds as XPointer/XLink proxies or special media players. However, as Web browsers and media players get more mature, developers are freed to concentrate on the interesting bit, namely providing users with powerful tools.

5.1 XLink

XLink (42) is a W3C recommendation for an XML based hypermedia format. XLink is a general XML format to describe navigational hypermedia, and to allow expressions of navigational hypermedia to be inserted into XML documents. While the main application of XLink is expected to be linking within XML documents, the standard itself is not limited to address solely XML locations, provided that appropriate locators have been defined. XLink supports simple links similar to the links currently found on the Web (i.e., unidirectional one-ary untyped links), as well as extended links, which can be bi-directional n-ary typed links stored externally to the constituent documents. An extended link consists of a number of `locator`s designating resources (a resource is the source or destination of a link—in HTML a local resource would be the text within an `<a></a>` tag). `Arc`s are used to describe transversal order in a link (i.e., from one location

to another). An interesting feature of XLink is the support for integration at the attribute level with other XML formats—through the use of a XLink name space an existing XML document can be extended to support XLink by adding a number of XLink attributes. This provides a relatively simple migration path for XML authors wishing to hypermedia enable their formats.

A number of XLink systems have been developed. Two proxy-based systems are Goate (43) and XLinkProxy (44). Both address the currently missing XPointer and XLink functionality in common Web browsers by adding and resolving XLinks at the proxy level. XLinkProxy provides an authoring interface implemented through DOM and Javascript, whereas Goate does not currently provide an authoring interface. A third system is Xspect (45), which is available both in a server- and client-side version. In addition to XLink, Xspect also supports navigational hypermedia and guided tours (which are visualised using Scalable Vector Graphics) as supported in OHIF (46), the interchange format used by Webvise and the Arakne Environment. While Xspect provides some authoring functionality, the system relies predominantly on Webvise for authoring. The OHIF file is subsequently converted to XLink through XSLT transformations.

XLink holds great potential. At this point the widespread Web browser support is still lacking, but if it should appear, it holds interesting prospects for the future of the Web. In a realised XLink future, complex linking structures are routinely stored outside of Web resources, immediately available for the users of the Web. In such a scenario, there will of course still be open hypermedia research. Indeed, just as the Web made "hypermedia" a widely understood term, XLink may have the same result for externally stored hypermedia structures.

5.2 Distribution and Scale

What lies in the future for the Arakne Environment and other Web augmentation tools? A widespread architecture for Web augmentation is yet to emerge. Systems such as Annotea (47) holds promise for annotations, and XLink (see Section 5.1) may well be the future format for external links on the Web. One issue that remains with these systems (as well as the other systems described in this paper) is that of Web-wide scalability. One architecture which has received much attention in recent years is peer-to-peer (P2P) networking. An architecture based on P2P could potentially scale well enough, and the next step for the Arakne Environment will be the investigation of whether such P2P infrastructure is a viable direction. Related issues are efficient (collaborative) filtering of links and annotations. Given the amount of noise found on the Internet in the form of spam etc. in email and elsewhere, a successful system must be able to provide its users with tools for filtering, moderation and verification.

6 Conclusion

The combination of open hypermedia and the Web has occasionally been contested (2), but the two worlds of the Web and open hypermedia have much to offer each other, as demonstrated by the Web augmentation tools described in this paper. Depending on setting and target application, the approach taken may vary, but in general the Web is a wonderfully extensible system, which allows for many sophisticated augmentations.

This paper has described a number of systems, which in one way or another augment the Web. The main contribution of the paper has been the general discussion of the possible approaches and associated trade-offs in the field of Web augmentation, as well as the description of the lessons learned through the development of several versions of the Arakne framework and the Arakne Environment.

Web augmentation allows users to structure their Web experience and to share this with other users of the Web. As a tool it can find use in both professional settings as a knowledge management and structuring tool, as well provide Web surfers, bloggers, or Web journalists with more powerful tools. The last few years have seen a proliferation of Web logs, many of which large report on occurrences on other Web sites. Web augmentation fits nicely within such a niche. Niches apart, Web augmentation tools cannot be considered truly successful, before they support the general user both in terms of ease of use and scalability. These are some of the greatest challenges ahead.

The Web is not rendered obsolete by Web augmentation, quite the contrary. Web augmentation strengthens the Web by extending the possibilities of interacting and using the Web. Ultimately, Web augmentation is a question of "liberating the link", and allowing users to participate in a free, Web-wide discourse by linking, associating, annotating, and restructuring the existing Web to fit their needs and vision.

Web augmentation as a technology is open hypermedia, which since 1989 has demonstrated its wide applicability. The unique advantage of Web augmentation as a subfield of open hypermedia, is that the Web in general works with open formats and standards. While proprietary differences in implementations exist and may hamper work, the principle of openness still holds. Compared to the general work of open hypermedia dealing with applications with often closely held proprietary document formats, the subfield of Web augmentation has it much easier. The possibilities inherent in the Web architecture (such as creating specialised hypermedia proxies), or the combination of dynamic HTML with scripting and style sheets are many indeed, as witnessed by the extensions and experiments done with the Arakne Environment as well as the many other Web augmentation tools described herein.

Acknowledgements

The author would like to thank the people who have contributed to Arakne over the years, both conceptual and code-wise: Kaj Grønbæk, Polle T. Zellweger, Jock Mackinlay, René Thomsen, Michael Bang Nielsen, and Henning Jehøj Madsen. The author would also like to thank the anonymous reviewers for many constructive suggestions.

References

1. BOUVIN, N. O. Unifying strategies for Web augmentation. In Tochtermann et al. (50), pp. 91–100.

2. NÜRNBERG, P. J., *and* ASHMAN, H. What was the question? reconciling open hypermedia and world wide web research. In Tochtermann et al. (50), pp. 83–90.

3. MILES-BOARD, T., CARR, L., *and* HALL, W. Looking for linking: Associative links on the web. In Anderson et al. (52), pp. 76–77.

4. ANDERSON, K. M. Supporting industrial hyperwebs: Lessons in scalability. In *Proceedings of the 21st International Conference on Software Engineering*, pp. 573–582, Los Angeles, CA, USA, May 1999.

5. MEYROWITZ, N. K. The missing link: Why we're all doing hypertext wrong. In BARRETT, E., editor, *The Society of Text: Hypertext, Hypermedia and the Social Construction of Information*, pp. 107–114. MIT Press, Cambridge, USA, 1989.

6. WHITEHEAD JR., E. J. An architectural model for application integration in open hypermedia environments. In Bernstein et al. (48), pp. 1–12.

7. HALL, W., DAVIS, H. C., *and* HUTCHINGS, G. *Rethinking Hypermedia: The Microcosm Approach*. Kluwer Academic Publishers, Norwell, USA, 1996.

8. GRØNBÆK, K., *and* TRIGG, R. H. Design issues for a Dexter-based hypermedia system. *Communications of the ACM*, 37(2):40–49, February 1994.

9. WIIL, U. K., *and* LEGGETT, J. J. The HyperDisco approach to open hypermedia systems. In *Proceedings of the 7th ACM Hypertext Conference*, pp. 140–148, Bethesda, MD USA, March 1996.

10. NÜRNBERG, P. J. *HOSS: An Environment to Support Structural Computing*. PhD thesis, Department of Computer Science, Texas A&M University, College Station, Texas, USA, 1997.

11. ANDERSON, K. M., TAYLOR, R. N., *and* WHITEHEAD, JR., E. J. Chimera: Hypermedia for heterogeneous software development environments. *ACM Transactions on Information Systems*, 18(3), July 2000.

12. CARR, L. A., ROURE, D. D., HALL, W., *and* HILL, G. The distributed link service: A tool for publishers, authors and readers. In *Proceedings of the 4th International World Wide Web Conference*, Boston, USA, December 1995. W3C.

13. CARR, L. A., HALL, W., *and* HITCHCOCK, S. Link services or link agents? In Grønbæk et al. (49), pp. 113–122.

14. EL-BELTAGY, S. R., HALL, W., ROURE, D. D., *and* CARR, L. Linking in context. In Davis et al. (51), pp. 151–160.

15. GRØNBÆK, K., BOUVIN, N. O., *and* SLOTH, L. Designing Dexter-based hypermedia services for the World Wide Web. In Bernstein et al. (48), pp. 146–156.

16. SANDVAD, E., GRØNBÆK, K., SLOTH, L., *and* KNUDSEN, J. L. A metro map metaphor for guided tours on the Web: the Webvise guided tour system. In *Proceedings of the 10th International World Wide Web Conference* (53), pp. 326–333.

17. ANDERSON, K. M. Integrating open hypermedia systems with the World Wide Web. In Bernstein et al. (48), pp. 157–166.

18. BOUVIN, N. O., ZELLWEGER, P. T., GRØNBÆK, K., *and* MACKINLAY, J. D. Fluid annotations through open hypermedia: Using and extending emerging Web standards. In *Proceedings of the 11th International World Wide Web Conference*, pp. 160–171, Honolulu, USA, May 2002. W3C.

19. BUSH, V. As we may think. *The Atlantic Monthly*, 176(1):101–108, July 1945.

20. VITALI, F., *and* BIEBER, M. Hypermedia on the Web: what will it take? *ACM Computing Surveys*, 31(4es):31–37, 1999.

21. HUGHES, G., *and* CARR, L. Microsoft smart tags: support, ignore or condemn them? In Anderson et al. (52), pp. 80–81.

22. FURUTA, R., SHIPMAN III, F. M., MARSHALL, C. C., D. BRENNER, D., *and* HSIEH, H.-W. Hypertext paths and the World-Wide Web: Experiences with Walden's Paths. In Bernstein et al. (48), pp. 167–176.

23. GRØNBÆK, K., SLOTH, L., *and* ØRBÆK, P. Webvise: browser and proxy support for open hypermedia structuring mechanisms of the World Wide Web. In *Proceedings of the 8th International World Wide Web Conference*, pp. 253–267, Toronto, Canada, May 1999. W3C.

24. BARRETT, R., *and* MAGLIO, P. P. Intermediaries: new places for producing and manipulating Web content. *Computer Networks and ISDN Systems*, (30):509–518, 1998.

25. BOUVIN, N. O. Designing open hypermedia applets: Experiences and prospects. In Grønbæk et al. (49), pp. 281–282.

26. BOUVIN, N. O., *and* SCHADE, R. Integrating temporal media and open hypermedia on the World Wide Web. *Computer Networks — The International Journal of Computer and Telecommunications Networking*, (31):1453–1465, 1999.

27. ZELLWEGER, P. T., BOUVIN, N. O., JEHØJ, H., *and* MACKINLAY, J. D. Fluid annotations in an open world. In Davis et al. (51), pp. 9–18.

28. WIIL, U. K., HICKS, D. L., *and* NÜRNBERG, P. J. Multiple open services: A new approach to service provision in open hypermedia systems. In Davis et al. (51), pp. 83–92.

29. JÜHNE, J., JENSEN, A. T., *and* GRØNBÆK, K. Ariadne: A Java-based guided tour system for the World Wide Web. In *Proceedings of the 7th International World Wide Web Conference*, Brisbane, Australia, April 1998. W3C.

30. REINERT, O., BUCKA-LASSEN, D., PEDERSEN, C. A., *and* NÜRNBERG, P. J. CAOS: A collaborative and open spatial structure service component with incremental spatial parsing. In Tochtermann et al. (50), pp. 49–50.

31. BOUVIN, N. O. Experiences with OHP and issues for the future. In *Proceedings of the Open Hypermedia Systems Workshop 6.0 Hypertext 2000*, number 1903 in Lecture Notes in Computer Science, pp. 13–22. Springer, 2000.

32. MARSHALL, C. C. Toward an ecology of hypertext annotation. In Grønbæk et al. (49), pp. 40–49.

33. HAAKE, J. M., *and* WILSON, B. Supporting collaborative writing of hyperdocuments in SEPIA. In *Conference proceedings on Computer-supported cooperative work*, pp. 138–146, Toronto, Canada, November 1992.

34. WIIL, U. K., *and* LEGGETT, J. J. Workspaces: The HyperDisco approach to Internet distribution. In Bernstein et al. (48), pp. 13–23.

35. REICH, S., WIIL, U. K., NÜRNBERG, P. J., DAVIS, H. C., GRØNBÆK, K., ANDERSON, K. M., MILLARD, D. E., *and* HAAKE, J. Addressing interoperability in open hypermedia: the design of the open hypermedia protocol. *The New Review of Hypermedia and Multimedia*, 5:207–248, 1999.

36. WHITEHEAD, JR., E. J. Uniform comparison of data models using containment modeling. In Anderson et al. (52), pp. 182–191.

37. PAGE, K. R., CRUICKSHANK, D., *and* ROURE, D. D. It's about time: link streams as continuous metadata. In Davis et al. (51), pp. 93–102.

38. CHANG, B.-W., MACKINLAY, J. D., ZELLWEGER, P. T., *and* IGARASHI, T. A negotiation architecture for fluid documents. In *Proceedings of the 11th annual ACM symposium on user interface software and technology*, pp. 123–132, San Francisco, CA USA, November 1998.

39. ZELLWEGER, P. T., CHANG, B.-W., *and* MACKINLAY, J. D. Fluid links for informed and incremental link transitions. In Grønbæk et al. (49), pp. 50–57.

40. 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 *Proceedings of the CHI 2000 conference on Human factors in computing systems*, pp. 249–256. ACM Press, 2000.

41. WEINREICH, H., OBENDORF, H., *and* LAMERSDORF, W. The look of the link—concepts for user interface of extended hypermedia. In Davis et al. (51), pp. 19–28.

42. DEROSE, S., MALER, E., ORCHARD, D., *and* TRAFFORD (EDITORS), B. XML Linking Language (XLink). W3C Recommendation 27 June 2001, W3C, June 2001. `http://www.w3.org/TR/xlink/`.

43. MARTIN, D., *and* ASHMAN, H. Goate: Xlink and beyond. In Anderson et al. (52), pp. 142–143.

44. VITALI, F., FOLLI, F., *and* TASSO, C. Two implementations of XPointer. In Anderson et al. (52), pp. 145–146.

45. CHRISTENSEN, B. G., *and* HANSEN, F. A. XLink—linking the Web and open hypermedia. In *Proceedings of the Open Hypermedia Systems Working Group Meeting 8.0*, pp. 9–18. FernUniversität Hagen, Germany, 2002.

46. GRØNBÆK, K., SLOTH, L., *and* BOUVIN, N. O. Open hypermedia as user controlled meta data for the Web. *Computer Networks*, (33):553–566, 2000.

47. KAHAN, J., KOIVUNEN, M.-R., PRUD'HOMMEAUX, E., *and* SWICK, R. R. Annotea: An open rdf infrastructure for shared web annotations. In *Proceedings of the 10th International World Wide Web Conference* (53).

48. BERNSTEIN, M., CARR, L., *and* ØSTERBYE, K., editors. *Proceedings of the 8th ACM Hypertext Conference*, Southampton, UK, April 1997. ACM Press.

49. GRØNBÆK, K., MYLONAS, E., *and* SHIPMAN, III, F. M., editors. *Proceedings of the 9th ACM Hypertext Conference*, Pittsburgh, PA USA, June 1998. ACM Press.

50. TOCHTERMANN, K., WESTBOMKE, J., WIIL, U. K., *and* LEGGETT, J. J., editors. *Proceedings of the 10th ACM Hypertext Conference*, Darmstadt, Germany, February 1999. ACM Press.

51. DAVIS, H., DOUGLAS, Y., *and* DURAND, D. G., editors. *Proceedings of the 12th ACM Hypertext Conference*, Århus, Denmark, August 2001.

52. ANDERSON, K. M., MOULTHROP, S., *and* BLUSTEIN, J., editors. *Proceedings of the 13th ACM Hypertext Conference*, College Park, Maryland, USA, June 2002. ACM Press.

53. W3C. *Proceedings of the 10th International World Wide Web Conference*, Hong Kong, May 2001.