

# XLink — Linking the Web and Open Hypermedia

*Bent Guldbjerg Christensen and Frank Allan Hansen*  
Department of Computer Science, University of Aarhus  
Åbogade 34, DK-8200 Aarhus N, Denmark  
Email: {bentor, fah}@daimi.au.dk

## ABSTRACT

This paper considers the use of XLink as a linking mechanism for both the Web and for open hypermedia systems. We present a comparison between the open hypermedia interchange format (OHIF) and XLink and describe a XLink implementation based on this comparison. Finally, we discuss whether XLink can bridge the waters between the Web and open hypermedia systems.

## KEYWORDS

XLink, OHIF, open hypermedia systems, World Wide Web

## INTRODUCTION

The World Wide Web (Web) is by far the most widely known and successful hypermedia system of today. It is used in almost every imaginable area from publishing to entertainment and trading. The success of the Web is probably due to its simple but extendable architecture. Web servers can be extended with CGI programs which allows for dynamically generated documents and Web browsers can be scripted with e.g. JavaScript or extended with various plug-ins and components. This flexibility makes the Web an ideal platform for a lot of applications.

The success of the Web has also resulted in the Web browser becoming a core part of almost any computing environment from desktop computers to PDAs and advanced cellular phones. This has the great advantage that the browser is ready at hand — to use the Web there is no need to install an extra application (which is often the case with other hypermedia systems).

But even though the Web is a versatile system it has a lot of shortcomings compared to other open hypermedia systems (OHSs) especially in the areas of hypermedia model, link- and structuring mechanisms and support for collaborative work.

With respect to link mechanisms the Web only supports very simple links. Links can only address whole documents and predefined anchors in the documents which makes finer grained linking impossible. Furthermore, both links and anchors are defined in-line in the documents so only the owner of a document can make links from the document. As a result the use of the Web is for most users a read-only experience. It also makes it hard for groups of people to share links or have

different collections of links attached to the same set of documents. The links supported by the Web are in essence simple go-tos. There is no support for more powerful relations such as bidirectional links, multi-headed links, or external out-of-line links. This lack of advanced link- and structuring mechanisms illustrates an area where the Web falls short compared to many OHSs.

The gap between the ubiquitous Web and OHSs has been discussed before [13], and work has been done to bridge the gap. One approach chosen by OHSs such as The Distributed Link Service [4], DHM [9], Chimera [1], Webwise [11], and Arakne [3] is to augment the Web with extra hypermedia functionality and thus provide the user of the Web with more powerful linking and structuring mechanisms. With this approach the Web is just treated as another client of the OHS. However, it has the disadvantage of introducing a new system component which is not a standard part of most computing environments in contrast to the Web browser.

Work is also being done in the Web community to improve the Web from within. The XLink recommendation from W3C [7] specifies a new linking mechanism for the Web which supports both the simple links used on the Web today, as well as more sophisticated links. However, we have found very few implementations that actually use XLink. It also appears that very little work has been done in investigating the qualities of XLink as a linking mechanism for the Web and its potential as linking mechanism for other systems e.g. OHSs (a preliminary investigation of XLink as an export format for Chimera was done by Halsey and Anderson in [12]).

In the remainder of this paper we will discuss our work with XLink as a linking format for the Web and for an OHS. We will describe our implementation of a set of XSLT stylesheets which in a very simple way makes it possible to do transformations between interchange files generated by the WebNize1000 OHS (a commercially available descendant of the Webwise [11] OHS) and XLink. The goal of our work is to investigate whether the use of XLink on the Web will reduce the gap described above and whether XLink is suitable as a linking format for OHSs. If the latter is the case XLink could be a great candidate for a common link format and thus increase the interoperability between OHSs and the Web.

## LINKING IN OPEN HYPERMEDIA SYSTEMS

Our investigation is of a comparative nature. We started out with an analysis of the data format used by the WebNize1000 OHS. WebNize supports bidirectional, multi-headed, span-to-span links which can be labeled with a name or description. It also supports global links which are similar to the generic links in Microcosm [6]. Furthermore, the system supports annotation of documents (or spans in a document) and the notion of Guided Tours which are trails of links through a set of documents. These elements are collected in a hypermedia context that is superimposed upon the documents by the system. With this format we feel we have a solid base for comparing the features of XLink with those of OHSs.

### THE OHIF FORMAT

The open hypermedia interchange format (OHIF) [10] was introduced together with the applications Webvise [11] and Arakne [3]. To define the OHIF format an XML DTD<sup>1</sup> was derived from the OHSWG navigational data model [8]. The key elements of the OHIF format are described shortly below, so a comparison with the transformed XLink version is possible.

`ohif:node`<sup>2</sup>: The `ohif:node` element is the fundamental hypermedia data object of OHIF. An `ohif:node` corresponds to a document and contains the URL to it.

`ohif:anchor`: An `ohif:anchor` element points out the location in an `ohif:node`'s content which is source or destination of a link. `ohif:anchor` elements contain a location specifier (`locSpec`) typically pointing to a text selection with a regular expression (a so called `simpleLoc`). `ohif:--annotations` are implemented as `ohif:anchors` with a presentation specifier that describes the type (`popup`, `replace`, `insert after`, `insert before`) and text of the annotation.

`ohif:endpoint`: An `ohif:endpoint` refers to an `ohif:--anchor` and holds a presentation specification (`PSpec`) which describes how the destination endpoint should be presented. `ohif:endpoints` also contains a direction attribute which defines whether the endpoint is source, destination, or both.

`ohif:link`: The `ohif:link` element contains a collection of `ohif:endpoint` references.

`ohif:guidedtour`: An `ohif:guidedtour` represents a graph of `ohif:nodes` and consists of two collections, one with `ohif:vertex ids` and one with `ohif:edge ids`. An `ohif:vertex` element refers to a hypermedia object typical an `ohif:node`. The `ohif:edge` element holds the ids for the source and the destination `ohif:vertex` of the `ohif:edge` and `PSpec` for the graphical presentation.

These are the fundamental elements of the OHIF format that are converted into a XLink based structure.

<sup>1</sup><http://www.daimi.au.dk/~les/ohif/ohif.dtd>

<sup>2</sup>To distinguish OHIF elements from XLink elements their names are prefixed with a XML namespace.

## XLINK

The XLink recommendation [7] describes a XML based linking format. XLink allows the expression of: multi-headed links, out-of-line links, and to associate meta data with a link. These are all well-known features of many open hypermedia systems, but what makes XLink special is that it is already a W3C standard.

XLink was originally designed to be the linking standard for XML documents and therefore has some XML specific properties, but the standard does not dictate how XLink elements should be used. The XLink elements can be used in several ways and with various perspectives. A XLink structure can be applied in-line to an existing XML data structure with the use of attributes inserted directly into the data elements. Used this way the linking information is considered a property of the data. The XLink structure can also be applied to the XML data by creating new XML elements that would only contain the linking information. In this way the XLink information is treated as first class data elements. A third and more interesting approach is to create a separate XML document that contains all of the linking information (a linkbase). All the links is thus out-of-line and the original data is not changed in any way.

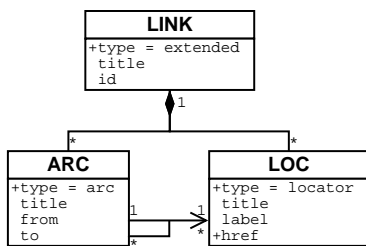
The XLink standard uses ordinary URI references to specify locations. When a more fine grained `locSpec` is wanted the URIs can be extended with a `XPointer` expression to identify URI fragments. The `XPointer` language [5] was constructed to support addressing into XML documents, but can be used for non XML data as well. A `XPointer` expression can be build up of sub expressions which are evaluated left to right until one of them succeeds. This fallback mechanism supports a fragment to be specified in several different ways to increase the chances of finding it. The expressions can include the use of functions defined by the `XPointer` standard in runtime calculations. Among these functions are a collection of simple string functions that we use to simulate the regular expressions used in OHIF based `locSpecs`.

### FROM OHS STRUCTURES TO XLINK

Our goal was to create a direct mapping between an OHIF file and a XLink linkbase. Therefore, we created mappings for each of the key OHIF elements described earlier. These mappings are presented below.

An `ohif:anchor` element and the `ohif:node` element that is used by the `ohif:anchor` are transformed to a XLink locator element named `xlink:loc`. The `xlink:loc` element keeps the URL from the `ohif:node` element and extends it with a `XPointer` expression that corresponds to the `ohif:anchor locSpec`.

The two `ohif:endpoint` elements that constitute a relation between two `ohif:anchors` are merged into one XLink arc element named `xlink:arc`. A `xlink:arc` element has two attributes `from` and `to` which hold ref-



**Figure 1: The structure of a `xlink:link` element. A `xlink:link` element can contain `xlink:loc`s that each specifies a location with a XPointer extended URI. The `xlink:arc` elements connects the `xlink:locs` to form links.**

ferences to the `xlink:loc` elements corresponding to the original `ohif:anchors`. The `xlink:arc` element can also contain PSpec information of how the destination of the link should be presented e.g. replacing the current view, a popup, or an in-line include.

### Links

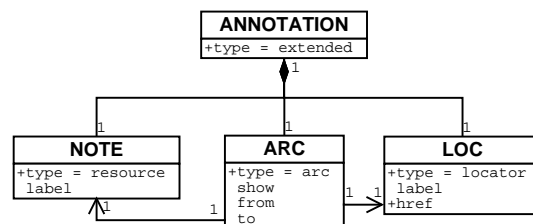
The `ohif:link` element is mapped to an element with the XLink type set to extended. The new `xlink:link` contains `xlink:arc` elements corresponding to `ohif:endpoints` and `xlink:loc` elements used by the `xlink:-arcs`. This link structure is depicted in figure 1.

The OHIF format supports the notion of global links. This can be expressed as a special variant of the `xlink:link` at figure 1. The only difference is that the `xlink:loc` elements which specify a source of a global link should only contain a XPointer expression and not the whole URI.

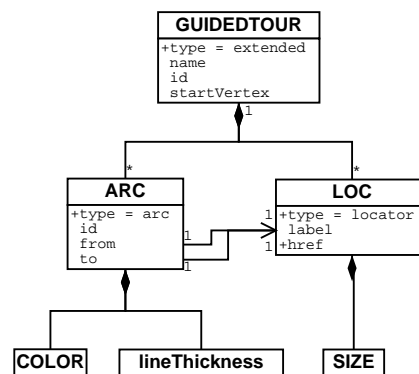
An `ohif:link` structure describing bi-directional links between two locations is listed in example 1. The corresponding `xlink:link` is listed in example 2. Notice that the `xlink:arcs` do not have an attribute named `to`. This is a shorthand for setting the `to` attribute to every `xlink:loc` element defined inside the current `xlink:link` element.

### Annotations

An annotation in OHIF is implemented as PSpecs to an `ohif:anchor`. In the XLink version an annotation is represented as a special case of the general link structure from figure 1. The information of the annotation is contained in the element named `xlink:annotation` which has the XLink type `extended`. The `xlink:annotation` element always contains the three elements: `xlink:loc`, `xlink:-arc`, and `xlink:note`. The `xlink:loc` element holds the presentation location of the annotation with a XPointer extended URI. The `xlink:note` element contains the actual annotation text. The `xlink:loc` and the `xlink:-note` elements are connected with the `xlink:arc` element which also describes how the annotation should be presented. The structure of a XLink annotation is presented in figure 2.



**Figure 2: The structure of a `xlink:-annotation`. The `xlink:loc` element specifies where the annotation should be presented. The actual annotation is contained in the `xlink:note` element. The `xlink:arc` element describes how the annotation should be presented.**



**Figure 3: The structure of a `xlink:-guidedtour`.**

### Guided tours

The `ohif:guidedtour` represents a graph of nodes. This is mapped into XLink structures with a `xlink:loc` element for each `ohif:vertex` and a `xlink:arc` element for each `ohif:edge`. The `ohif:guidedtour` includes PSpecs for both `ohif:vertex` and `ohif:edge` elements (coordinates, color, size, ...) these are preserved as sub elements of both the `xlink:loc` and `xlink:arc` elements. The structure of a `xlink:guidedtour` can be seen in figure 3. Notice how the structure of the `xlink:-guidedtour` is very similar to that of a `xlink:link`.

### XSPECT – A SIMPLE IMPLEMENTATION OF XLINK

Our implementation, the Xspect system<sup>3</sup>, consists of a set of XSLT stylesheets which realize the transformation between OHIF and XLink linkbases. We have also created stylesheets that transform the XLink linkbases into a HTML and JavaScript representation that can be used directly in standard Web browsers. The transformation system is illustrated in figure 4.

<sup>3</sup>A demo of the system can be found on the URL: <http://www.daيمي.au.dk/~fah/xspect/start.html>

---

**Example 1:** A two-headed bi-directional ohif:link.

---

```
<LINK id="daimi.4.1017075150" name="Link 4">
<ENDPOINTIDSET>
<ID>daimi.7.1017075150</ID><ID>daimi.24.1017079993</ID>
</ENDPOINTIDSET>
</LINK>

<ENDPOINT id="daimi.7.1017075150" name="AARHUS"
  linkid="daimi.4.1017075150" anchorid="daimi.6.1017075150" direction="BIDIRECTIONAL" >
<PSPECIDSET><ID>daimi.5.1017075150</ID></PSPECIDSET>
</ENDPOINT>

<ANCHOR id="daimi.6.1017075150" parentid="daimi.3.1017075150">
<SIMPLELOC occurrence="1" >
<SELECTION>AARHUS</SELECTION>
<SELECTIONCONTEXT>UNIVERSITY OF AARHUS</SELECTIONCONTEXT>
</SIMPLELOC>
</ANCHOR>

<NODE id="daimi.3.1017075150" name="Welcome to Computer Science in Aarhus (DAIMI)">
<CONTENTSPEC version="" mimetype="application/WWWAddress" >
<PROPERTIES>
<PROPERTY name="docTitle" type="System" flags="0">
<VALUESET>
<VALUE>Welcome to Computer Science in Aarhus (DAIMI)</VALUE></VALUESET>
</PROPERTY>
</PROPERTIES>
<URL>http://www.daimi.au.dk/</URL>
</CONTENTSPEC>
</NODE>

<ENDPOINT id="daimi.24.1017079993" name="Department" linkid="daimi.4.1017075150"
  anchorid="daimi.23.1017079993" direction="BIDIRECTIONAL" >
<PSPECIDSET><ID>daimi.22.1017079993</ID></PSPECIDSET>
</ENDPOINT>

<ANCHOR id="daimi.23.1017079993" parentid="daimi.21.1017079993">
<SIMPLELOC occurrence="1" >
<SELECTION>Department</SELECTION>
<SELECTIONCONTEXT>About the Department</SELECTIONCONTEXT>
</SIMPLELOC>
</ANCHOR>

<NODE id="daimi.21.1017079993" name="About the Department">
<CONTENTSPEC version="" mimetype="application/WWWAddress" >
<PROPERTIES>
<PROPERTY name="docTitle" type="System" flags="0">
<VALUESET>
<VALUE>About the Department</VALUE></VALUESET>
</PROPERTY>
</PROPERTIES>
<URL>http://www.daimi.au.dk/doc74.html</URL>
</CONTENTSPEC>
</NODE>
```

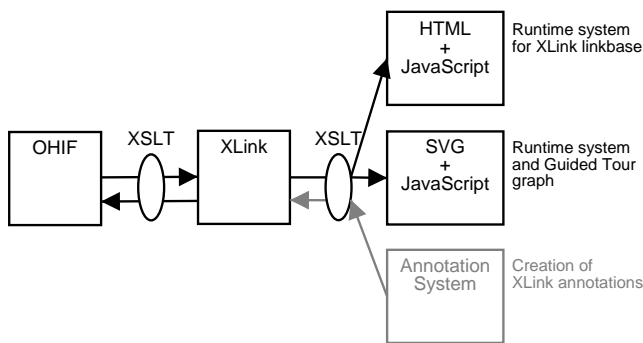
---

**Example 2:** A two-headed bi-directional xlink:link. This xlink:link corresponds to the ohif:link in example 1.

---

```
<LINK xlink:type="extended" xlink:title="Link 4" xlink:id="daimi.4.1017075150">
<ARC xlink:type="arc" xlink:title="AARHUS" xlink:from="daimi.6.1017075150"/>
<LOC xlink:type="locator" xlink:label="daimi.6.1017075150"
  xlink:href="http://www.daimi.au.dk/#xpointer(string-range(/,&quot;UNIVERSITY OF AARHUS&quot;,15,6)[1])"
  xlink:title="Welcome to Computer Science in Aarhus (DAIMI)"/>
<ARC xlink:type="arc" xlink:title="Department" xlink:from="daimi.23.1017079993"/>
<LOC xlink:type="locator" xlink:label="daimi.23.1017079993"
  xlink:href="http://www.daimi.au.dk/doc74.html#xpointer(string-range(/,&quot;About the\
  Department&quot;,11,10)[1])"
  xlink:title="About the Department"/>
</LINK>
```

---



**Figure 4: The Xspect systems transformation architecture.**

Furthermore, the transformation system is closed in that transformation from XLink to OHIF is supported. This indicates a structural equivalence between the two formats.

We use the XSLT stylesheets in two implementations: a client only version implemented in Microsoft's Internet Explorer and a CGI server version.

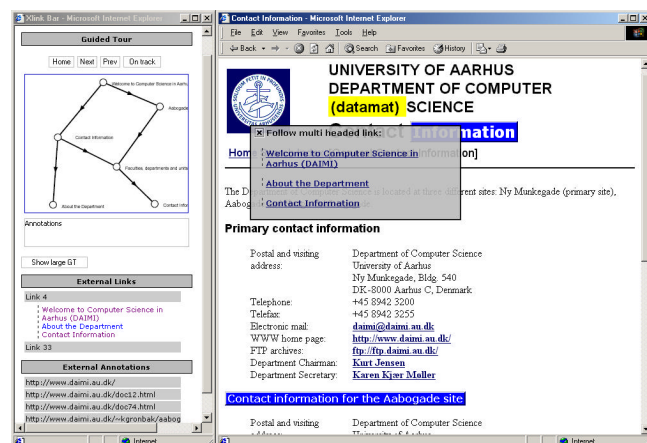
The client version uses Microsoft's XSLT processor to translate the OHIF contexts into XLink and further into HTML and JavaScript that is displayed in the browser. JavaScript is used to implement the runtime representations of the XLink linkbase and to decorate documents with anchors and link- and annotation dialogs.

The server version uses a Python based XSLT processor to do the same transformation as mentioned above. The server also handles decoration of documents by altering the HTML code, so this is not done in JavaScript as in the client version. Furthermore, the server implements transformation from XLink to SVG which is used to display the Guided Tours as interactive metromaps. The metromaps are scripted with JavaScript to make them function similarly to the metromaps in the WebNize OHS.

Besides these transformations we have also implemented an annotation system in the client version. The annotation system allows users to select a span of text in a HTML document and annotate it. The annotation is then inserted in a global annotation XLink linkbase that can be used by the Xspect system or transformed to OHIF and used by WebNize. A session in the Xspect system is illustrated in figure 5.

## DISCUSSION AND FUTURE WORK

We are now ready to discuss our experiences with XLink. First of all, is XLink a suitable link mechanism for the Web? We definitely think so! XLink offers vast improvements over the simple links used on the Web today. With respect to linking, XLink brings the Web much closer to other OHSs. However, only a limited number of applications natively supports XLink at the moment (the Amaya browser from W3C and the



**Figure 5: A session in the Xspect system. The left window displays a complete context with a Guided Tour, links, and annotations. The right window displays a document decorated with link- and annotation anchors. The user has activated a link and the corresponding link dialog is open.**

Mozilla browser implement simple links, but not extended links). If XLink is to be the linking mechanism for the next generation Web we will have to see more complete support for XLink and also more widespread support in other mainstream applications.

Can XLink be used as linking mechanisms in OHSs? We have succeeded in implementing a direct mapping from OHIF to XLink so in this case XLink is a suitable format. The biggest difference between the two formats is that OHIF uses a referential organization of its elements while elements in XLink are organized into aggregated structures. In our implementation we did not find this to be a problem. Another important point to note is that XLink is not just for linking XML. As an example, the Xspect system is used to link HTML documents. Our use of text based XPointer expressions makes it possible to address the exact same things that are addressable by the OHIF simpleLocs. Furthermore, XPointer is especially suited for the format of the fragment identifiers used within the URI references when XML documents are being linked, but the XLink specification does not require locators to be XPointer based. Thus, when linking none XML documents it is possible to employ other types of fragment identifiers suitable for the specific document type.

Finally, can the adoption of XLink be part of the bridge between OHSs and the Web? As discussed above we think XLink is suitable for linking in both the Web and in OHSs. Furthermore, the Xspect system is an example of an implementation where XLink is used as an interchange format between the structures of OHIF and the structures of a Web application. We find such an approach important in the effort of bridging the gap between OHSs and the Web. However, this approach also raises some issues. As described earlier,

XLink can be applied to data in a variety of ways: both as attributes of the data and also as first class data elements. If XLink is to be used as a middleware format between OHSs and the Web we need to agree on a way to do this. We would suggest a catalog of *best practices* or a collection of design patterns on how XLink can be applied to applications in a way suitable for both Web- and OHS applications. We think that such a catalog could be useful for both the open hypermedia community and the Web community and it would also provide some examples of the use of XLink which are seriously lacking at the moment.

In our work with XLink linkbases issues regarding the support of collaborative work and scalability were considered. Both issues could possibly be overcome by fragmenting linkbases into appropriate chunks. These linkbase parts should be of a size convenient for locking mechanisms like Web-DAV [14]. Fragmentation would also support the structuring of massive link information creating the notion of cascading linkbases. The XLink recommendation specifies a fragmentation mechanism that probably could be used. This is an area we intend to investigate further.

## CONCLUSION

We think that XLink holds great promise — both as a higher level linking mechanism for the next generation Web but also as linking mechanism for other applications.

But before we will see this happening there are some issues that remain to be solved. The Xspect system uses XSLT transformations to convert XLink linkbases to HTML and JavaScript that can be accessed by conventional browsers. This approach proved to be both simple and powerful but on a larger scale this method is cumbersome. If XLink is to be widely accepted and used, we need native support in mainstream applications (e.g. browsers and editors). We also see the need of some common guidelines on how to employ XLink since this can be done in a variety of different ways. This is important if XLink indeed is to become the standard way of linking.

## ACKNOWLEDGEMENT

We would like to thank Niels Olof Bouvin for his good advice and comments on improving earlier versions of this paper.

## REFERENCES

1. Kenneth M. Anderson. Integrating open hypermedia systems with the World Wide Web. In Bernstein et al. [2], pages 157–166.
2. Mark Bernstein, Leslie Carr, and Kasper Østerbye, editors. *Proceedings of the 8<sup>th</sup> ACM Hypertext Conference*, Southampton, UK, April 1997.
3. Niels Olof Bouvin. Unifying strategies for Web augmentation. In Tochtermann et al. [15], pages 91–100.
4. Leslie A. Carr, David De Roure, Wendy Hall, and Gary Hill. The distributed link service: A tool for publishers, authors and readers. In *Proceedings of the 4<sup>th</sup> International World Wide Web Conference*, Boston, USA, December 1995.
5. Ron Daniel, Steve DeRose, and Eve Maler (editors). XML Pointer Language (XPointer). W3C candidate recommendation, W3C, September 2001. <http://www.w3.org/TR/xptr>.
6. Hugh C. Davis, Simon Knight, and Wendy Hall. Light hypermedia link services: A study of third party integration. In *Proceedings of the 1994 ACM European conference on Hypermedia technology*, pages 41–50, Edinburgh, UK, September 1994.
7. Steve DeRose, Eve Maler, David Orchard, and Ben Trafford (editors). XML Linking Language (XLink). W3c recommendation, W3C, June 2001. <http://www.w3.org/TR/xlink/>.
8. Kaj Grønbaek. OHS interoperability — issues beyond the protocol. In *Proceedings of OHS Workshop 4.0 held at Hypertext '98, Pittsburgh, June 20-24, 1998*.
9. Kaj Grønbaek, Niels Olof Bouvin, and Lennert Sloth. Designing Dexter-based hypermedia services for the World Wide Web. In Bernstein et al. [2], pages 146–156.
10. Kaj Grønbaek, Lennert Sloth, and Niels Olof Bouvin. Open hypermedia as user controlled meta data for the Web. In *Proceedings of the 9<sup>th</sup> International World Wide Web Conference*, pages 553–566, Amsterdam, Holland, May 2000.
11. Kaj Grønbaek, Lennert Sloth, and Peter Ørbæk. Webvise: browser and proxy support for open hypermedia structuring mechanisms of the World Wide Web. In *Proceedings of the 8<sup>th</sup> International World Wide Web Conference*, pages 253–267, Toronto, Canada, May 1999.
12. Brent Halsey and Kenneth M. Anderson. XLink and open hypermedia systems: A preliminary investigation. In *Proceedings of the 11<sup>th</sup> ACM Hypertext Conference*, pages 212–213, San Antonio, TX USA, May 2000.
13. Peter John Nürnberg and Helen Ashman. What was the question? reconciling open hypermedia and world wide web research. In Tochtermann et al. [15], pages 83–90.
14. Greg Stein. Webdav resources. <http://www.webdav.org/>.
15. Klaus Tochtermann, Jörg Westbomke, Uffe K. Wiil, and John J. Leggett, editors. *Proceedings of the 10<sup>th</sup> ACM Hypertext Conference*, Darmstadt, Germany, February 1999.