Xspect: Bridging Open Hypermedia and XLink

Bent G. Christensen
Dept. of Computer Science
University of Aarhus
Aabogade 34, 8200 Århus N, Denmark
bentor@daimi.au.dk

Frank Allan Hansen
Dept. of Computer Science
University of Aarhus
Aabogade 34, 8200 Århus N, Denmark
fah@daimi.au.dk

Niels Olof Bouvin
Dept. of Computer Science
University of Aarhus
Aabogade 34, 8200 Århus N, Denmark
n.o.bouvin@daimi.au.dk

ABSTRACT

This paper evaluates the XLink format in comparison with other linking formats. The comparison is based on Xspect, an implementation of XLink. Xspect handles transformation between an open hypermedia format (OHIF) and XLink, and the paper discusses this isomorphic transformation and generalises it to include another open hypermedia format, FOHM. The Xspect system[1], based on XSLT and Javascript, provides users with an interface to browse and merge linkbases. Xspect supports navigational hypermedia in the form of links inserted on the fly into Web pages, as well as guided tours presented as SVG. Xspect has two implementations: one server-side and one running on the client. Both implementation provide the user with an interface for the creation of annotations.

The main result of the paper is a critique of XLink. XLink is shown to be a format well suited for navigational hypermedia, but lacking in more advanced constructs. More problematic are the issues regarding large-scale use, such as evaluating validity and credibility of linkbases, and ensuring general support for a format as flexible as XLink.

Categories and Subject Descriptors

H.5.4 [**Information Interfaces and Presentation**]: Hypermedia

General Terms

XLink, Open Hypermedia

Keywords

XLink, XPointer, SVG, OHIF, FOHM, Xspect, Open Hypermedia, annotations

1. INTRODUCTION

The Web as it stands today is a remarkable success, which, in part, can 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 simplicity of HTML links (in-lined, uni-directional, 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 disjoint sets of links are needed on the same body of documents, the simple HTML link does not suffice. This can be illustrated by considering support for critical reading. There may be a lot of information available on the Web, but users are not free to annotate these pages as they can with a book. Related is the proliferation of Web journalism and meta Web sites, largely consisting of references (with short introductions) to pages on other Web sites. Common to these cases is that the creators of comments are limited in referring to other Web pages: they can either make whole page references, or copy/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, to creating bi-directional multi-headed links etc. These activities may be termed "Web augmentation" [4] 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 to Web pages). For a general discussion of this subject, see [4]. The more exotic systems have suffered from a lack of shared standards, which has severely limited adoption, inter-operation and interchange.

Trying to address some of the problems with the current Web and doing this with common standards, W3C has developed a linking format, XLink [12], and a meta-data format, RDF [23]. As W3C recommended formats, these technologies may make advanced hypermedia widely available on the Web and elsewhere. We describe in this paper an implementation of XLink, Xspect, based on standard tools (XSLT [1] and Javascript). Xspect uses XLink, but can convert to and from other hypermedia formats. Apart from navigational hypermedia, Xspect provides a SVG [19] interface to guided tours and an authoring environment for annotations. The implementation of Xspect leads to a comparison between XLink and other hypermedia formats, as well as a general critique of XLink. Among the contributions of the paper is the discussion of transformations needed to convert to and from XLink and other hypermedia formats.

The paper is structured as follows: Section 2 introduces related work done with hypermedia formats and systems; Section 2.3 delves deeper into hypermedia formats; Section 3 describes the Xspect prototype; Section 4 reports on the experiences with Xspect and formulates a critique of XLink; Section 5 describes future work, and the paper is concluded in section 6.

2. RELATED WORK

This section relates the work done in the open hypermedia com-

[1]The Xspect system can be found at `http://fahbentor.daimi.au.dk`

WWW2003, May 20–24, 2003, Budapest, Hungary.
ACM 1-58113-680-3/03/0005.

munity and in the Web community with a special focus on systems offering linking or annotation support. In the context of the work described in this paper, a distinction must be made between systems implementing some hypermedia functionality and the data formats used by the systems, as the full generality of a format is rarely fully implemented. Both aspects will be covered below.

2.1 Open Hypermedia Systems on the Web

Since 1995 the open hypermedia community have been developing various systems for augmenting the Web with externally stored hypermedia structures. Walden's Paths [15] supports trails along Web pages using CGI on a Web server. Ariadne [20] is an applet-based approach to provide guided tours.

One of the earliest systems to support linking was the Distributed Link Service [8], which provided users with generic links on Web pages. DLS was later developed into AgentDLS [7] and a number of other systems. A recurring element in most of these implementation has been the use of a proxy server to modify Web pages or to provide links (and other meta-data).

Other systems include Webvise [16, 28] and the Arakne Environment [4, 5]. Both systems rely on the integration with a Web browser (Microsoft Internet Explorer) for modification of Web pages (e.g., inserting links and annotations).

These systems have relied on their own data formats, such as FOHM [26] or OHIF [16], which are described in Section 2.3.

2.2 XLink Based Systems

XLink has been the basis of some implementations. These include xlinkit [27], Goate [24], and XLinkProxy [31].

xlinkit is a rule based system primarily designed for integrity checks in a distributed XML document set, though it can also be used to generate links between XML elements. A user writes a number of rules or constraints in an XML based rule language, and the system generates an XLink linkbase linking elements meeting or violating the specified constraints. The resulting XLink linkbase can subsequently be combined with the XML documents and rendered into, e.g., HTML by the system's XLink processor, XtooX.

Goate provides advanced hypermedia functionality through the use of a Web proxy. XLink is supported through a module which inserts links and destinations into HTML documents through the proxy. To help users differentiate between original and added links, the system marks the added links in different background colours.

XLinkProxy is similar to Goate in that it provides support for XLink linkbases through the use of a Web proxy.

A number of systems have sought to support annotations on the Web. Of special interest in a Web standards context is Annotea [21]. Annotations are RDF statements made by an author about a Web document. Annotations are external to the documents and can be stored in one or more annotation servers, e.g., implemented as a general RDF database. Annotea has been implemented in the W3C Amaya Web browser, through the use of bookmarklets, and in Annozilla [3], an extension to the Mozilla Web browser. Annotea annotations are represented via a combination of RDF [23], Dublin Core [14], XPointer [10], and XLink [12].

Annotations are (in Amaya) marked with a pencil icon on the location pointed out by the a:context attribute in the source document. The annotation text is stored either locally or on a server as a separate HTML file (the a:body resource). At runtime the annotation is spawned in a new browser window with the Dublin Core attributes shown in a table before the annotation text.

See [5, 33] for further discussion of annotations and sophisticated markup thereof in a Web context.

2.3 Hypermedia Formats

The notion of a common format for hypermedia interchange is by no means new. One of the stated goals of the Dexter Hypertext Reference Model [18] was the creation of an interchange format to facilitate inter-operation between hypermedia systems (this goal was never fully realised, though it formed a basis for an interesting discussion on the topic of hypermedia interchange [22]). Other efforts include HyTime [13]. While most hypermedia systems undoubtedly have the ability of dumping the contents of a link database into a text file in some format, few systems have used published formats for their use. This section describes the XLink format and two hypermedia formats developed by the open hypermedia community, the Fundamental Open Hypermedia Model (FOHM) [26] and the Open Hypermedia Interchange Format (OHIF) [16]. We will in this section focus more on the formats themselves rather than how they are handled by current implementations.

2.3.1 XLink

XLink [12] 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., uni-directional, 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). Arcs are used to describe traversal order in a link (i.e., from one location to another). Two examples of extended XLink links are illustrated in Figure 1.

An interesting feature of XLink is the support for integration at the attribute level with other XML formats—through the use of an XLink namespace an existing XML document can be extended to support XLink through the addition of a number of XLink attributes. This provides a relatively simple migration path for XML authors wishing to add hypermedia to their XML documents.

While XLink is quite open with regards to the document types that can be addressed, it uses XPointer [10] to identify regions of interest in XML documents. XPointer allows for selection based on IDs, hierarchical structure (from XPath [9]), or an arbitrary user selection (e.g., selecting a string in the rendered XML document). A given region may be identified using several locators, which improves reliability, as one locator might fail after a document has been edited. Combined with XPointer and XPath, XLink is well suited for linking XML documents.

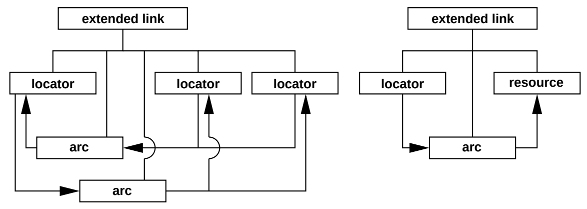


Figure 1: Two XLink links. The link on the left is multi-headed and bi-directional. The link on the right is uni-directional linking a local resource to a remote resource.

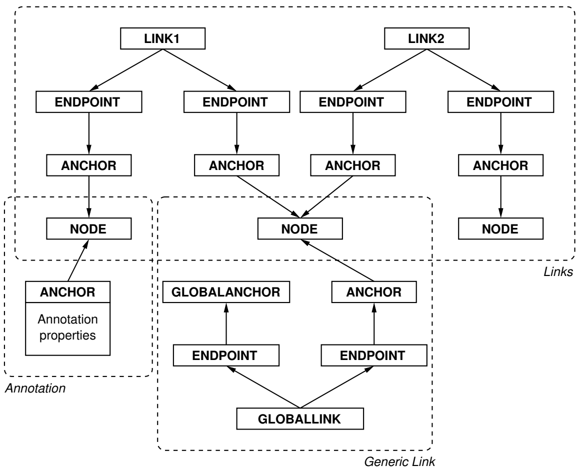


Figure 2: An OHIF context with two multi-headed links, a generic link, and an annotation.

2.3.2 *OHIF*

The Open Hypermedia Interchange Format (OHIF) [16] is supported by Webvise [28] and the Arakne Environment [5]. OHIF is formally defined by a DTD[2] derived from the Open Hypermedia Systems Working Group[3] (OHSWG) navigational data model [11]. In addition to navigational hypermedia (i.e., links) OHIF supports composites and guided tours. The constituent elements of the format share certain attributes, i.e., they all have globally unique identifiers (GUIDs), arbitrary key/value pairs, and associated presentation specifications (PSpecs). A consequence of the use of GUIDs is that references are used extensively rather than containment. In OHIF a `node` element corresponds to a document or resource, typically represented by a URL. URLs are wrapped in `node`s to support reuse (multiple `anchors` in the same document can use the same `node` as shown in Figure 2) and ease maintainability (if a Web page is moved, only one `node` should be updated). An `anchor` element points out the location in a `node`'s content which is source or destination of a `link`. `anchor` elements contain a location specifier (LocSpec) [17] typically identifying a text selection with a regular expression. `Annotation`s are implemented as `anchor`s with a PSpec that describes the type (popup, replace, prefix, postfix) and text of the annotation. An `endpoint` refers to an `anchor` and contains a direction attribute which defines whether the endpoint is source, destination, or both. `Endpoint`s are interconnected by a `link`. In the general case, `anchor`s can be associated with any hypermedia object (`link`, `endpoint`, `anchor`, and `node`) making it possible to link to links or annotate annotations. A `guidedtour` represents a graph of `node`s and consists of two collections, one with `vertex` IDs and one with `edge` IDs. A `vertex` element refers to a hypermedia object (e.g., a `node`). The `edge` element holds the IDs for the source and the destination `vertex` of the `edge` and PSpec for the graphical presentation.

2.3.3 *FOHM*

The link server Auld Linky [25] supports an XML data format based on the Fundamental Open Hypermedia Model (FOHM). Auld Linky implements mainly the navigational subset of the structures expressible in the general FOHM model. The FOHM format `association` elements describe the relationship between `data` elements. The relationship can e.g., be a link or a tour. `Data` elements may either be wrappers for remote resources (like OHIF `node`s) or contain the resources in-line themselves. Anchors within the resources wrapped by `data` elements are defined inside `reference` elements with e.g., a character offset for resources containing text. Each `data` element is contained inside a `reference` element (representing the anchors) which again is contained inside a `binding` element. The `binding` element is included in an `association` element to bind anchors to an association. A `binding` element describes whether the anchors are sources or destinations in a link structure or in which sequence the anchors are organised in a tour structure. A link structure can in this way be described by a single `association` element containing all needed link information. Furthermore, FOHM supports references between the same kind of first-class elements (`association`, `binding`, `reference`, and `data`). This feature provides easy reuse of e.g., a `data` element with multiple anchor definitions (as depicted in Figure 3). An advanced feature implemented in Auld Linky is the ability to denote in which "context" `binding` elements (and thereby the anchors defined in them) are accessible for the end user. The user's "context" (or state) is stored in the client and can be changed dynamically in a browsing session according to `behaviour` elements, which are child elements of bindings.

In the following section the open hypermedia formats OHIF and FOHM are compared to the XLink format. Since the Xspect prototype is based on OHIF transformations we will primarily focus on the mapping from OHIF to XLink. However, since both FOHM and OHIF are derived from the work on the OHSWG navigational data model and the formats to a wide extent are structurally equivalent, the mappings apply to FOHM as well, unless otherwise noted.

[2]`http://www.daimi.au.dk/~les/ohif/ohif.dtd`
[3]`http://www.cs.aue.auc.dk/ohswg/`

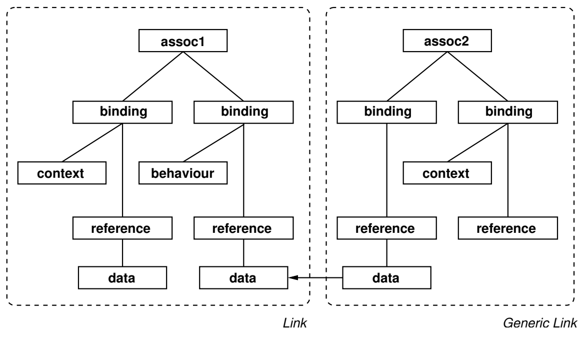


Figure 3: A multi-headed link and a generic link in FOHM.

3. XSPECT

The Xspect prototype discussed in this section is an XLink implementation built to demonstrate several aspects of the XLink linking language. The prototype implements isomorphic transformations between OHIF structures and corresponding XLink link structures. In this way, Xspect demonstrates which open hypermedia concepts can be represented with the XLink syntax. Furthermore, our discussion of the prototype illustrates the limitations of XLink and where the mappings between OHIF and XLink break down.

The Xspect transformation system also illustrates the possibilities of using XLink as an interchange between different systems e.g., between open hypermedia systems that support the same kind of functionality but express it in different formats such as OHIF and FOHM. The interchange can, however, also take place between open hypermedia systems and the Web. Hence, XLink may not only bring advanced linking mechanisms to the Web but can also be a way to bridge the worlds of open hypermedia and the Web.

Finally, the prototype exemplifies how the XLink technology can be utilised with existing standard technologies available in common Web browsers such as XSLT, SVG, Javascript, HTML, CSS, etc.

3.1 From OHS Structures to XLink

Before discussing the Xspect prototype implementation, the mappings between OHIF structures and XLink structures as used in the transformation implementation are outlined.

3.1.1 Links

The OHIF format supports the expression of bi-directional, *n*-ary, typed, out-of-line links. The links are organised in a referential structure with a link element definition and a number of associated endpoints. Each endpoint holds a reference to an anchor element that specifies the endpoint's location in a document which in turn is represented by a node element (see Figure 2).

The endpoints define the traversal behaviour of the link by means of a direction attribute which specifies whether an endpoint is a source, a destination, or both. The link can be traversed from sources to destinations. Endpoints also hold a presentation specifier (PSpec) with information on how the endpoint's node should be presented when the link is traversed. The endpoints' locations are specified by a LocSpec in the anchor element. OHIF supports several kinds of LocSpecs; the most commonly used is the SimpleLoc which specifies a location in a node along two axes—by a text selection and its occurrence in the document text and by the selection context which specifies some text surrounding the selection. The use of two axes improves endpoint location robustness even if changes occur in the document. The node elements hold a normal URL reference to the linked documents. Thus, the nodes can be regarded as document wrappers or document proxies. An example of a link in the OHIF format can be seen in Figure 4.

XLink was designed to support this kind of functionality, so a mapping from OHIF links to XLink links is straight forward. However, the structure of the two link types differs on several points. XLink defines links in a single `extended`-type element container and not as a referential structure. Anchoring is defined by URI references with a suitable fragment identifier, so the role of the OHIF node and anchor elements are merged into a single attribute value of `locator`-type elements. Furthermore, the concepts of link anchors and link traversal have been separated in XLink. Thus, there are no link "sources" or link "destinations". Instead a link is made up of a number of participating resources (specified by locators) and a number of traversal routes (specified by arcs) between the resources. This traversal scheme is more flexible than the OHIF direction specification, since a resource may be the starting point or ending point for one traversal route without having the same function for all other traversal routes in the link.

The mapping from OHIF links to XLink links can be divided into anchoring and traversal. XLink uses URI references to specify resources. The URIs can be extended with fragment identifiers for finer grained location specification. The adjunct XPointer standard is a natural candidate for the format of the fragment identifiers, since XPointer can be used with URIs to specify resources or parts of resources that will participate in a link. XPointer is an application of XPath and can thus specify element sets and arbitrary text strings in an XML document, but XPointer goes beyond XPath in that it can select text strings crossing multiple nodes which is exactly what we need to map the OHIF SimpleLocs to XLink.

SimpleLocs specifies the location of an endpoint by the occurrence of a text selection and its surrounding context. With the XPointer `string-range()` function [10, sec. 5.4.2] we can simulate this mechanism. Consider this XPointer expression

```
xpointer(string-range(/,'selection text')[3])
```

This expression selects the third occurrence of the string "selection text" in any text node in the document. However, this only includes one of the axes in the SimpleLoc. The second axes, the selection context, can be added in the following way:

```
xpointer(string-range(/,'Text surrounding\
              the selection text',22,14)[3])
```

Now the expression contains the selection context as well. This expression selects the third occurrence of the substring that is fourteen characters long and starts at the 22nd character: "selection text". Thus, we are back were we started. We need a way to specify both of the axes at the same time. This can be done by constructing the XPointer with more than one *XPointer part*

```
xpointer(string-range(/,'selection text')[3])\
    xpointer(string-range(/,'Text surrounding\
                         the selection text'))
```

The expression is evaluated from left to right and the intent of the second XPointer part is that it is an approximation of the first. Hence, we do not have an exact simulation of the SimpleLoc, but have created an XPointer with a fall-back mechanism. First the third occurrence of the selection text is searched for. If this fails any occurrence of the selection context is searched for.

When combined with the URL of a document this type of XPointer expressions can be used as values for the `href`-attribute in `locator`-type XLink elements to specify link endpoints. This does,

```
<LINK id="daimi.15" type="LINK" name="Example Link">
 <ENDPOINTIDSET>
  <ID>daimi.17</ID><ID>daimi.19</ID>
 </ENDPOINTIDSET>
</LINK>

<ENDPOINT id="daimi.17" type="ENDPOINT" name="EndPoint 17"
          linkid="daimi.15" anchorid="daimi.16"
          direction="BIDIRECTIONAL">
</ENDPOINT>

<ENDPOINT id="daimi.19" type="ENDPOINT" name="EndPoint 19"
          linkid="daimi.15" anchorid="daimi.18"
          direction="BIDIRECTIONAL">
</ENDPOINT>

<ANCHOR id="daimi.16" type="ANCHOR" parentid="daimi.14">
 <SIMPLELOC occurrence="1" >
  <SELECTION>substring 1</SELECTION>
  <SELECTIONCONTEXT>
   String including substring 1.
  </SELECTIONCONTEXT>
 </SIMPLELOC>
</ANCHOR>

<ANCHOR id="daimi.18" type="ANCHOR" parentid="daimi.14">
 <SIMPLELOC occurrence="1" >
  <SELECTION>substring 2</SELECTION>
  <SELECTIONCONTEXT>
   String including substring 2.
  </SELECTIONCONTEXT>
 </SIMPLELOC>
</ANCHOR>

<NODE id="daimi.14" type="NODE" name="Example node">
 <URL>http://www.some_domain.dk</URL>
</NODE>
```

Figure 4: An example of a bi-directional link in OHIF.

however, only specify the collection of associated resources that participate in the link, but does not specify the traversal amongst the resources. Traversal is defined by arcs that specifies traversal routes between labelled locators (or labelled local resources). So, the behaviour of an OHIF link can be mapped to XLink by creating a locator for each endpoint element and specifying a suitable URL and XPointer expression that simulates the endpoint's associated anchor and node elements. If the endpoint is a source of the link we will label the locator with a "src" label, if it is a destination we will label the locator with a "dest" label and so fourth. All that remains is then to create routes from every "src" labelled locator to every "dest" labelled locator:

```
<link xlink:type="extended">
 <loc xlink:type="locator"
      xlink:href="..."
      xlink:label="src"/>

 <loc xlink:type="locator"
      xlink:href="..."
      xlink:label="dest"/>

 <loc xlink:type="locator"
      xlink:href="..."
      xlink:label="both"/>
  ...
 <arc xlink:type="arc" xlink:from="src" xlink:to="dest"/>
 <arc xlink:type="arc" xlink:from="src" xlink:to="both"/>
 <arc xlink:type="arc" xlink:from="both" xlink:to="both"/>
 <arc xlink:type="arc" xlink:from="both" xlink:to="dest"/>
</link>
```

This construction specifies an XLink link with the same behaviour as an OHIF link—a link involving multiple resources and with multiple traversal routes. Note that since there can be multiple locators with the same label the arc specification actually creates multiple arcs that can be traversed at the same time.

3.1.2 Generic Links

OHIF also supports the specification of another link type: the generic link (or global link in OHIF terminology). The generic link is conceptually somewhat different from normal links. It specifies a single fixed endpoint and a number of generic endpoints that are applied *globally* to any document. The generic endpoints are thus not associated with a specific document, but contains an anchor specification consisting of a text string that can be used as the starting point for the link. When the link is activated it is traversed from a generic endpoint to the location specified by the fixed endpoint.

Generic links can, e.g., be used as cross references in dictionaries or technical documentation where the generic endpoints specifies words and the fixed endpoint specifies the location of the words' definition. Generic links can also be representations of queries: the generic endpoints could specify words from the query and the fixed endpoint would be the result of the query.

OHIF global links are specified by a normal destination endpoint with associated anchor and node elements, and a number of source endpoints with references to `globalanchor` elements (see Figure 2). These elements do not have associated note elements, but specify just a text string in a SimpleLoc. If an OHIF context includes multiple definitions of identical SimpleLocs in different global links (the links have identical generic endpoints), the links will be presented as a multi-headed global link when activated from these endpoints.

The OHIF global links can be mapped to XLink by creating a single fixed locator as in the preceding section with a URL and XPointer fragment identifier. This locator will function as the fixed endpoint in the link so we will give it a "dest" label. Each of the generic endpoints is mapped to a locator which only contains an XPointer expression and thus specifies a generic fragment identifier. These locators should have a "src" valued label, since they will be used as sources of the link. Finally, we can create a single arc going from the generic locators to the fixed locator:

```
<link xlink:type="extended"
    xlink:role="http://fahbentor.daimi.au.dk/globallink">

 <loc xlink:type="locator"
      xlink:href="..."
      xlink:label="dest"/>

 <loc xlink:type="locator"
      xlink:href="#xpointer(string-range(/,'str1'))"
      xlink:label="src"/>

 <loc xlink:type="locator"
      xlink:href="#xpointer(string-range(/,'str2'))"
      xlink:label="src"/>
  ...
 <arc xlink:type="arc" xlink:from="src" xlink:to="dest"/>
</link>
```

However, since no "base" document has been defined for the generic locators the associated XPointer fragment identifiers will normally be evaluated relative to the context in which they are defined i.e., the linkbase. This is not the intended behaviour, since the locators should be applied globally to any document. Thus, we need a way to specify this intention to the XLink application.

The XLink standard provides three semantic attributes for specifying the meaning of links: the `title`, the `role`, and the `arcrole`. The title attribute is normally used to describe the meaning of a link in a human-readable form. The role and arcrole attributes always contain a URI which references some meta-data that describes the meaning of the link. The role attribute is intended to describe the meaning of the link element while the intended use of the arcrole is to describe the relationship of the linked resources. However, if we discard the use of general meta-data, the role and arcrole attributes can be used for link type declarations. In the example above we have included a role attribute in the `extended`-type element with a URI that functions as a type declaration for global links. The link now has a semantic type that can be recognised and interpreted as a global link by an XLink aware application. But since there is no recommended or agreed upon standard for XLink link types each application that interprets this kind of links should be programmed to recognise some fixed set of link roles.

3.1.3 Annotations

In the process of critical reading or in settings where several people cooperate on the work of a set of documents, the ability to add annotations to selected parts of the material is equally important to the ability to structure the resources by links. In fact, one of the earliest visions of hypermedia systems, proposed by Vannevar Bush [6], included the possibility of adding comments and annotations to the material stored in his proposed Memex device in much the same way it would have been done in a physical book.

Annotations in OHIF are specified by an anchor and a node element identifying the location of the annotation object in a document. Furthermore, the anchor element contains a set of properties defining the annotation type and its content (see Figure 2). OHIF supports several types of annotation: pop-up annotations work somewhat like footnotes, replacing annotations replaces a text range (specified by the anchor's LocSpec) in the document with the content of the annotation, and prefix and postfix annotations are embedded before or after a text range in the document. Hence, they resemble hand-written nodes directly in the text.

The OHIF annotations can be represented as XLink links by utilising the XLink `resource`-type element. XLink resource elements define local resources that participate in a link and since they may have any content, we can use them to hold the text of the user comments. By creating an arc going from the annotated document to the local resource we get an in-bound link associating the annotation with the document.

However, annotations are not documents but rather user created meta-data associated with documents. Consequently, they should not behave or be treated as documents. The XLink standard allows behaviour attributes in arc elements that specifies how and when resources in a link should be presented.

The `actuate` attribute is intended to describe the timing of link traversal. The attribute must have one of the values "onLoad", "onRequest", "other", or "none". Clearly, the "onRequest" value is what we want for the pop-up annotations, since they often contain secondary information which is related to the document, but is not a part of it. The embedded and replacing annotation types on the other hand often contains information that should be read in the context of the document text and they should therefore be inserted when the document is loaded. So, the "onLoad" value could be used with these annotation types.

The `show` attribute describes the presentation of the ending resource (the annotation in this case) and must have one of the values "new", "replace", "embed", "other", or "none". These values are suited for our purpose—when dealing with a pop-up annotation we could use the "new" value, a replacing annotation could use the "replace" value, and the prefix and postfix types could be simplified using the "embed" value. Moreover, the "other" value specifies that the application should look for other markup present in the link specifying the presentation. If we wanted to support more advanced annotation such as fluid annotations [5], the use of the "other" value would allow for advanced (not XLink specific) presentation specifications.

When the behaviour attributes are combined with the described link construction, the result is an XLink annotation that closely simulates the OHIF annotation. To illustrate this, consider the following link fragment that specifies a pop-up annotation:

```
<link xlink:type="extended"
    xlink:role="http://fahbentor.daimi.au.dk/annotation">

 <loc xlink:type="locator"
      xlink:href="..."
      xlink:label="src"/>

 <note xlink:type="resource" xlink:label="dest">
   annotation text
 </note>

 <arc xlink:type="arc"
      xlink:from="src"
      xlink:to="dest"
      xlink:actuate="onRequest"
      xlink:show="new"/>
</link>
```

Even though this construction includes the specification of the desired presentation we are still facing that general purpose XLink applications are likely to treat the resources in the link as documents. Hence, we have again typed the link by means of a specific URI in the role attribute to indicate that this is an annotation link.

3.1.4 Guided Tours

The final OHIF hypermedia structure we will discuss is the guided tour. As described in section 2 numerous systems have been developed to support human-created tours, trails, or paths through a set of related information resources.

The OHIF guided tour does not only describe how the resources are structured but includes a representation of graphical *metro maps* representing the structure. Each stop in a map represents an information resource such as a Web page and can be annotated with both text and graphics describing the resource. The map outlines one or more *routes* which the user can follow and thereby be guided through the information collection (for further information on the metro map metaphor, see [28]).

OHIF does not describe guided tours by links, but uses instead a referential composite structure which contains a number of vertex and edge elements. These elements describe the appearance and behaviour of nodes and edges constituting a directed graph that is used as the basis for the metro map. Each vertex element also has an associated node element that identifies the location of a resource in the same way as nodes in the OHIF links.

It is quite easy to map the *structure* of a guided tour to an XLink link. For each vertex and node pair we can create a locator that identifies the same resource. The routes represented by edges in the tour can be represented by corresponding arcs in the link. Consider the following link fragment:

```
<link xlink:type="extended"
    xlink:role="http://fahbentor.daimi.au.dk/gt">

 <loc xlink:type="locator"
      xlink:href="..."
      xlink:label="v1"/>

 <loc xlink:type="locator"
      xlink:href="..."
      xlink:label="v2"/>

 <loc xlink:type="locator"
      xlink:href="..."
      xlink:label="v3"/>

 <arc xlink:type="arc" xlink:from="v1" xlink:to="v2"/>
 <arc xlink:type="arc" xlink:from="v2" xlink:to="v3"/>
 <arc xlink:type="arc" xlink:from="v1" xlink:to="v3"/>
</link>
```

The link describes a guided tour containing three nodes and two routes: a route from *v1* to *v2* to *v3*, and a route from *v1* to *v3*.

However, this only specifies the participating resources and how to traverse between them but contains no specification on how to present the resources. As discussed above, XLink allows for specification of how ending resources should be presented when the link is traversed but this is not what we need here—rather we need a way to specify how the link should be presented as a whole! The problem can be approached in several ways. We could add the presentation specification as "PSpec" child elements of the XLink link elements i.e., locators and arcs could have child elements specifying how the corresponding nodes and edges in the graph should be represented. However, such elements would have no XLink specific meaning and would thus just be an application specific add-on to XLink. Another approach could be to combine XLink with another XML format which could hold the presentation of the tour. SVG could e.g., be used to express the graphical layout and the XLink structures could be applied to the SVG elements. But this would lead to a guided tour expressed in SVG rather than in XLink.

Matters get worse when considering other structures. FOHM allows first class elements to be reused in different parts of the hypermedia structure. Similarly, OHIF supports anchors pointing into other hypermedia objects e.g., links. Since XLink provides no concept of link structures inside link structures these FOHM and OHIF constructions are very hard to express in XLink. By the same token, if the concept of guided tours is generalised to include tabletops (as in NoteCards [30]), where each stop in a guided tour is not just a representation of a single document but of a set of documents and their properties, we face similar difficulties, since XLink does not define the meaning of linking to links or of composites.

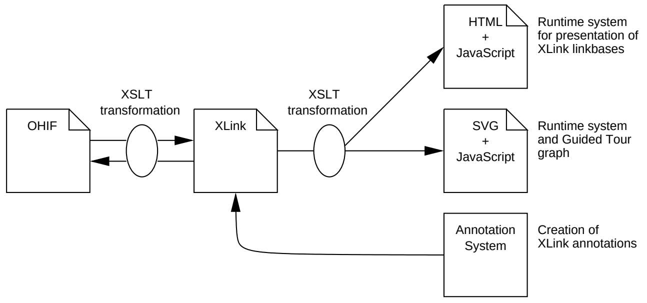

XSLT transformation
XSLT transformation
OHIF
XLink
HTML + JavaScript
Runtime system for presentation of XLink linkbases
SVG + JavaScript
Runtime system and Guided Tour graph
Annotation System
Creation of XLink annotations

Figure 5: The transformation architecture of Xspect.

This discussion leads to an important issue regarding XLink. As the examples have illustrated, XLink is quite adequate when describing navigational hypermedia structures. Indeed, the mappings presented here illustrate how XLink can be used to express a number of different structures. However, when facing other structuring mechanisms such as the OHIF guided tour composite where the collection of resources as a whole is the key factor, we have found it difficult to express the structures with XLink syntax alone.

3.2 The Xspect prototype

Xspect is an XLink based hypermedia system. The system consists of a set of XSLT stylesheets describing the transformation from OHIF files to XLink linkbases and back to OHIF files again. Furthermore, the system contains stylesheets which transform the XLink linkbases to a HTML and Javascript representation that may be delivered directly to standard Web browsers. The transformation architecture of the Xspect system is illustrated in Figure 5.

We use the XSLT stylesheets in two implementations: a client version using Microsoft Internet Explorer and a CGI server version.

3.2.1 The Xspect Client Version

The client version uses Microsoft's XSLT processor to translate OHIF contexts into XLink linkbases. The linkbases are transformed further into a HTML and Javascript runtime representation of the linkbases which are displayed in the browser. The implementation illustrates merging XLink linkbases into HTML documents and doing so with standard tools provided by the browser without requiring extra components to utilise the XLink technology.

We use the linking abilities provided by HTML and Javascript as a "low-level" linking language to construct the "high-level" XLink structures as conceptually introduced by the Goate system [24]. This includes the dynamic decoration of documents with anchors, link menus, and annotation dialogs. However, the implementation choice of only utilising standard Web technologies lead to the "same origin policy" restriction enforced by the browser on the Javascript routines. The "same origin policy" prevents documents or scripts loaded from one origin from getting or setting properties of a document from a different origin. This restricts the client version to manipulate HTML documents only from the same domain as the server with the Javascript files.

The architecture of the client version is depicted in Figure 6. Notice that the linked HTML documents is completely separated from the XLink linkbases. Only when used by the Xspect system the linkbase structures are merged into the HTML documents.

3.2.2 The Xspect Server Version

The Xspect server version uses a Python based XSLT processor. The same transformations as described above are performed, but

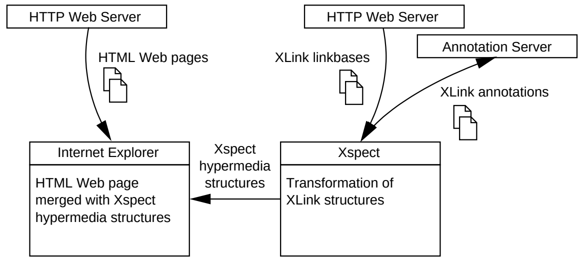

HTTP Web Server
HTTP Web Server
Annotation Server
HTML Web pages
XLink linkbases
XLink annotations
Internet Explorer
HTML Web page merged with Xspect hypermedia structures
Xspect hypermedia structures
Xspect
Transformation of XLink structures

Figure 6: Xspect client implementation. External XLink structures are merged with the linked HTML pages in the browser.

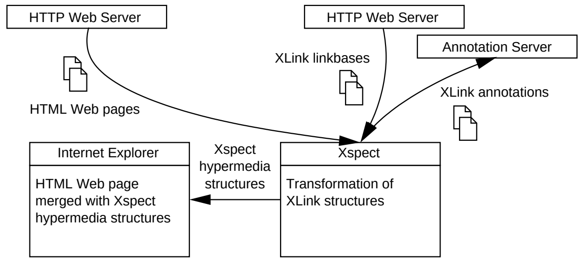

HTTP Web Server
HTTP Web Server
Annotation Server
XLink linkbases
HTML Web pages
XLink annotations
Internet Explorer
HTML Web page merged with Xspect hypermedia structures
Xspect hypermedia structures
Xspect
Transformation of XLink structures

Figure 7: Xspect server implementation. The external XLink structures are merged with the linked HTML pages on the server and then delivered to the browser.

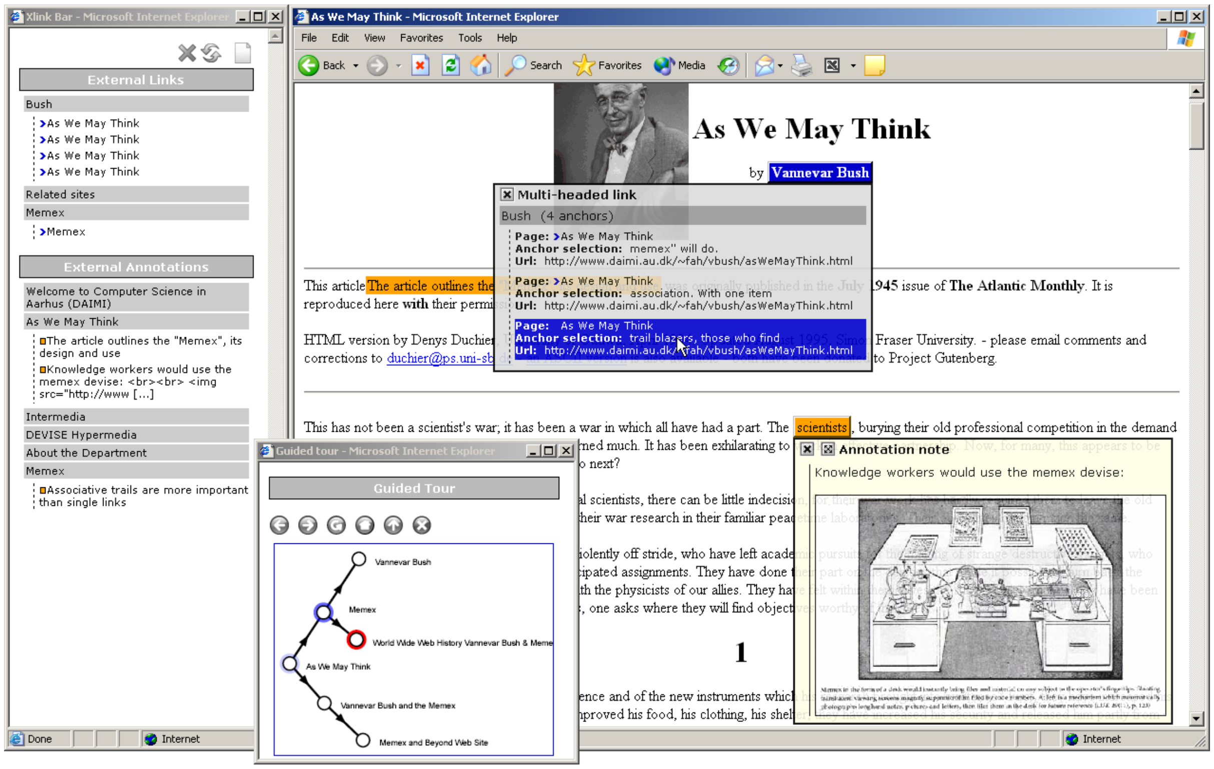
the processing is executed at the server instead of in the browser. The output from the server processing is a HTML and Javascript representation of the document with the appropriate anchors inserted. This eliminates the "same origin policy" that occurs in the client version, because both the Javascripts and the HTML documents are delivered by the same server. The server also implements a transformation from XLink to SVG that is used when the linkbase contains a guided tour. The SVG version of a guided tour is displayed as an interactive metro map scripted with Javascript.

Figure 7 illustrates the server version architecture. All computation regarding the merging of linkbase structures with the linked HTML document is performed at the server and afterwards delivered to the browser. Again, the XLink link structures are completely separated from the linked documents until processed by the Xspect system.

3.2.3 The Annotation System

Both versions also include an annotation system. This system allows a user to select a span of text in a HTML document and annotate it. The annotation is then inserted into either a personal linkbase or a global linkbase on a dedicated server. The linkbases created this way can be used by the Xspect system or transformed to OHIF and used by an OHIF aware client. Merging of linkbases is also supported. This may be used to merge the global annotation linkbase with another linkbase of choice. The following scenario illustrates some of the possibilities with linkbase merging:

Jim attends a course in Hypermedia and is studying a paper by Vannevar Bush. Jim has some questions regarding the paper, so he annotates the paper with them. The other members of his study group merges Jim's annotations and try to answer his questions. In this way the members of the study group discuss and help each other to understand the paper through annotations. But Jim still has some unclear points which the study group could not help him



Xlink Bar - Microsoft Internet Explorer

External Links

Bush
- As We May Think
- As We May Think
- As We May Think
- As We May Think

Related sites

Memex
- Memex

External Annotations

Welcome to Computer Science in Aarhus (DAIMI)

As We May Think
- The article outlines the "Memex", its design and use
- Knowledge workers would use the memex devise:

 <img src="http://www [...]

Intermedia

DEVISE Hypermedia

About the Department

Memex
- Associative trails are more important than single links

As We May Think - Microsoft Internet Explorer

As We May Think

by Vannevar Bush

Multi-headed link

Bush (4 anchors)

Page: As We May Think
Anchor selection: memex" will do.
Url: http://www.daimi.au.dk/~fah/vbush/asWeMayThink.html

Page: As We May Think
Anchor selection: association. With one item
Url: http://www.daimi.au.dk/~fah/vbush/asWeMayThink.html

Page: As We May Think
Anchor selection: trail blazers, those who find
Url: http://www.daimi.au.dk/~fah/vbush/asWeMayThink.html

This article The article outlines the ... 1945 issue of The Atlantic Monthly. It is reproduced here with their permi...

HTML version by Denys Duchier ... Fraser University. - please email comments and corrections to duchier@ps.uni-sb ... to Project Gutenberg.

This has not been a scientist's war; it has been a war in which all have had a part. The scientists, burying their old professional competition in the demand ...ned much. It has been exhilarating to ... o next?

Annotation note

Knowledge workers would use the memex devise:

Guided tour - Microsoft Internet Explorer

Guided Tour

Vannevar Bush
Memex
World Wide Web History Vannevar Bush & Meme
As We May Think
Vannevar Bush and the Memex
Memex and Beyond Web Site
Done
Internet

Figure 8: A session in the Xspect system. The left window displays an overview of a linkbase with links and annotations. The right window displays a Web page decorated with link and annotation anchors. The user has activated a pop-up annotation and a multi-headed link and the corresponding link dialog is open. On top of the two windows floats a small window with a guided tour.

with. Therefore Jim contacts the teacher who merges the annotations of all the group members in Jim's group to read their discussions and she adds an explanation to Jim's questions.

A session in the Xspect system showing Jim's work after annotating the paper is illustrated in Figure 8. Besides annotations the Xspect system currently also supports visualisation and functionality of multi-headed and bi-directional links and guided tours. Examples of these structures are also shown in Jim's session.

The scenario presents how a linkbase can be used as an information layer upon the original document. Merging of linkbases is implemented with an XSLT stylesheet. The stylesheet compares the `extended`-type elements of the two input linkbases, and if the same `extended`-type element is found in both input linkbases this element is only included once in the output linkbase. Since each `extended`-type element contains all the information associated with a link, such elements may be inserted into or removed from a linkbase without affecting other `extended`-type elements in the linkbase. This is not the case for link elements in either the OHIF or FOHM format, because the link structures in these formats may be highly fragmented and connected by references. Both OHIF and FOHM format files have a "global" state compared to the "local" state of XLink linkbases, because of the reuse of elements and references. Thus, the whole OHIF or FOHM file needs to be parsed to resolve the link structures.

Merging linkbases just creates additional layers on the document. But the merging also introduces problems with identifying the original source, author, and credibility of information retrieved from a linkbase composed from other linkbases. This issue is currently not addressed by the XLink recommendation. A possible solution to this issue of credibility of link information is discussed in the section.

4. RESULTS

This section describes noteworthy characteristics of the Xspect prototype and the current XLink standard.

The Xspect prototype illustrates two major results. First: Xspect implements a two-way transformation system between open hypermedia formats and XLink. The transformations illustrates the power of the XLink format and its structural equivalence with open hypermedia data formats. Second: Xspect is build solely on Web standards. Hence, it demonstrates a Web-standard-only implementation of open hypermedia and illustrates how XLink can bring the techniques of open hypermedia to the Web.

As the previous sections have demonstrated, XLink is a powerful and usable standard for navigational hypermedia. In Xspect, we were able create mappings between the structures used in open hypermedia formats and equivalent XLink structures. However, issues arose around the simulation of the OHIF LocSpecs which we were only able to approximate in XPointer. But since XLink is not limited to the use of XPointer as the scheme for fragment identifiers, this issue is not inherent to XLink but rather to XPointer. However, the OHIF and FOHM formats are not limited to structuring by links but also support structuring by composition. We found it hard to express structures such as links within links, guided tours, and tabletops using XLink syntax alone.

The approach taken in the implementation of the Xspect prototype illustrates another noteworthy issue. From the perspective of open hypermedia, the transformation architecture of Xspect can be seen as a vehicle for publishing open hypermedia structures to the Web. From the perspective of Web and XLink applications, Xspect can be viewed as a middleware component making it feasible to use the tools developed for open hypermedia systems as authoring environments for XLink structures (which is indeed what we have done in the development of the prototype). We find the latter perspective very important since no tools exist which fully and natively support the creation of XLink structures in work-like settings.

The support for on-the-fly merging of linkbases in the prototype introduced an issue of how to determine the credibility of information retrieved from a linkbase that is composed from other linkbases. XLink linkbases may be fragmented by the level of a single `extended`-type element, and should thus at this level contain credibility information such as: author, time of creation, and version. As previously described the purpose of the XLink `title`, `role`, and `arcrole` attributes is to associated meta-data with a link. But this meta-data is intended to describe the meaning and perspective of the link and the linked resources and we believe that credibility information is not that kind of meta-data. A solution may be to introduce new XLink attribute types dedicated to provide credibility information which could follow standards such as the Dublin Core [14].

Another issue involving linkbases is how to use them. The XLink standard suggests that linkbases are used to manage collections of related linking elements. Once the linkbases are created, they can be associated with specific resources by specifying a link with an arc specification containing a special arcrole. When the link is traversed, the arc specification will instruct the XLink application to load the linkbase and extract its links rather than doing normal processing. As we see it, XLink linkbases used this way only serve a very limited purpose since the content of the linkbase has to be known in advance when associated with a specific resource for the loading of the linkbase to have any effect. This approach is very document-centric and is similar to the concept of separating style information from the content of a document by use of external CSS-stylesheets. However, it does indeed support the separation of links from the linked resources, which has been one of the key concepts in the work of open hypermedia. The Xspect system takes another approach to the handling of linkbases. In principle each resource associated with an XLink linkbase is described in the linkbase itself by means of locators. Therefore, the Xspect system loads a linkbase and processes resources whenever they are loaded *and* have associated links in the linkbase. Hence, no previous knowledge about the linking elements in the linkbase is needed. We find this approach to be more link- and use-centric than the approach suggested by the standard.

The XLink standard is intended to offer a standardised way to do linking. It is formulated as an "enabling" XML vocabulary, which is used with other XML vocabularies to provide linking. As the discussion in this paper has illustrated, this has led to a very general linking mechanism adequate for many applications. It would, however, in most cases suffice with a fixed set of agreed-upon structures (e.g., links, generic links, annotations, etc.). Such a set would also play a crucial role, if XLink is to be used as the common interchange format between, e.g., different open hypermedia system and between these systems and Web applications. We believe that the creation of a catalog of *best practices* which outlines what structures are commonly needed and how these structures should be specified. This could result in a normative DTD for this XLink application or in a set of well-defined roles for link elements that could serve as types for the structure set. This would resemble the situation that we find with other XML technologies. For instance, XPath is the syntax used to describe parts of an XML document. XPath is designed very generic so it can be used as the basis for other applications such as XPointer and XSLT. XLink could thus be seen as the general linking specification that supported more specific applications suitable for, e.g., the Web and open hypermedia. We propose the Xspect system as one such best praxis and as a part of the discussion on how to utilise the XLink technology.[4] Furthermore, when considering the lessons learned in the development of the Web it seems that simple and well-defined standards attract more development activity than more complex and hard to understand technologies. We think that well-defined guidelines on how to use XLink in specific applications could foster the development of native XLink support in mainstream applications (e.g., browsers and editors) which is seriously lacking at the moment.

5. FUTURE WORK

One of the clear limitations of the current prototype implementation is the "same origin restrictions" imposed on the Javascript routines used to locate and modify Web pages. The server version of the Xspect system overcomes this limitation by imposing proxy-like functionality on the documents, but for many applications concerned with performance and availability this approach is not perfect. Ideally, this functionality should not be custom built, but provided by the Web browser through XLink support.

In the current implementation, the Xspect system allows users to load and browse linkbases. This means that the same linkbase can be viewed and sorted in different ways and that links can be browsed in ways not specified by arcs (e.g., a link with a single locator and no arcs can be presented through browsing of the linkbase). However, we have not investigated the possibilities of querying and searching linkbases. Such work would make it possible to search the content of links and annotations defined in large shared linkbases and to compose new structures e.g., generic links, from the links defined in the linkbases.

6. CONCLUSION

We have through our implementation and analysis demonstrated that XLink is a viable and flexible format for representing navigational hypermedia. Based solely on Web technologies, the Xspect prototype implements open hypermedia techniques based on XLink. The two-way transformation system of Xspect underlines the equivalence between XLink and open hypermedia data formats. The structural equivalence of the formats opens the possibilities of using open hypermedia systems as authoring environments for XLink based systems since support for XLink is still lacking in standard tools such as editors and browsers.

Our work identifies issues on representing structuring by composition with the linking mechanisms supported by XLink. Hypermedia structures such as links within links, guided tours, and tabletop composites are hard to express in meaningful ways with XLink.

Further issues arose around the use of credibility information when merging external linkbases. We argue that credibility information should be present at the link level of linkbases and that the set of semantic attributes provide by XLink could be expanded to support this explicitly.

A general problem of XLink is ironically its flexibility, as it opens the doors for wildly different implementation, which may or may not inter-operate. This paper calls for the creation of a catalog of best practices on how to apply XLink in ways suitable for Web and open hypermedia applications.

[4] A DTD derived from our experiences with Xspect can be found at `http://fahbentor.daimi.au.dk/xspect.dtd`

7. REFERENCES

[1] S. Adler, A. Berglund, J. Caruso, S. Deach, T. Graham, P. Grosso, E. Gutentag, A. Milowski, S. Parnell, J. Richman, and S. Zilles. Extensible stylesheet language (XSL). W3c recommendation, W3C, Oct. 2001. `http://www.w3.org/TR/xsl/`.

[2] K. M. Anderson, S. Moulthrop, and J. Blustein, editors. *Proc. of the 13th ACM Hypertext Conf.*, College Park, Maryland, USA, June 2002. ACM Press.

[3] Annozilla. `http://annozilla.mozdev.org/`.

[4] N. O. Bouvin. Unifying strategies for Web augmentation. In Tochtermann et al. [29], pages 91–100.

[5] N. O. Bouvin, P. T. Zellweger, K. Grønbæk, and J. D. Mackinlay. Fluid annotations through open hypermedia: Using and extending emerging Web standards. In *Proc. of the 11th WWW Conf.*, pages 160–172, Honolulu, USA, May 2002. W3C.

[6] V. Bush. As we may think. *The Atlantic Monthly*, 176(1):101–108, July 1945.

[7] L. A. Carr, W. Hall, and S. Hitchcock. Link services or link agents? In K. Grønbæk, E. Mylonas, and F. M. Shipman, III, editors, *Proc. of the 9th ACM Hypertext Conf.*, pages 113–122, Pittsburgh, PA, USA, June 1998.

[8] L. A. Carr, D. D. Roure, W. Hall, and G. Hill. The distributed link service: A tool for publishers, authors and readers. In *Proc. of the 4th WWW Conf.*, Boston, USA, Dec. 1995.

[9] J. Clark and S. DeRose (editors). XML Path language (XPath) version 1.0. W3C Recommendation 16 November 1999, W3C, Nov. 1999. `http://www.w3.org/TR/xpath`.

[10] R. Daniel, S. DeRose, and E. Maler (editors). XML Pointer Language (XPointer). W3c candidate recommendation, W3C, Sept. 2001. `http://www.w3.org/TR/xptr`.

[11] H. C. Davis, D. E. Millard, S. Reich, N. O. Bouvin, K. Grønbæk, K. M. Anderson, U. K. Wiil, P. J. Nürnberg, and L. Sloth. Interoperability between hypermedia systems: The standardisation work of the OHSWG. In Tochtermann et al. [29], pages 201–202.

[12] S. DeRose, E. Maler, D. Orchard, and B. Trafford (editors). XML Linking Language (XLink). W3C Recommendation 27 June 2001, W3C, June 2001. `http://www.w3.org/TR/xlink/`.

[13] S. J. DeRose and D. G. Durand. *Making Hypermedia Work: A User's Guide to HyTime*. Kluwer Press, Boston, 1994.

[14] Dublin Core metadata element set, version 1.1: Reference description. DCMI Recommendation 1999-07-02, DCMI, July 1999. `http://dublincore.org/documents/dces/`.

[15] R. Furuta, F. M. Shipman III, C. C. Marshall, D. D. Brenner, and H.-W. Hsieh. Hypertext paths and the World-Wide Web: Experiences with Walden's Paths. In M. Bernstein, L. Carr, and K. Østerbye, editors, *Proc. of the 8th ACM Hypertext Conf.*, pages 167–176, Southampton, UK, Apr. 1997.

[16] K. Grønbæk, L. Sloth, and N. O. Bouvin. Open hypermedia as user controlled meta data for the Web. *Computer Networks*, (33):553–566, 2000.

[17] K. Grønbæk and R. H. Trigg. Toward a Dexter-based model for open hypermedia: Unifying embedded references and link objects. In *Proc. of the 7th ACM Hypertext Conf.*, pages 149–160, Bethesda, MD, USA, Mar. 1996.

[18] F. G. Halasz and M. Schwartz. The Dexter hypertext reference model. *Communications of the ACM*, 37(2):30–39, Feb. 1994.

[19] D. Jackson and J. Ferraiolo. Scalable Vector Graphics (SVG) 1.1 Specification. W3C recommendation, W3C, Jan. 2003. `http://www.w3.org/TR/SVG11/`.

[20] J. Jühne, A. T. Jensen, and K. Grønbæk. Ariadne: A Java-based guided tour system for the World Wide Web. In *Proc. of the 7th WWW Conf.*, Brisbane, Australia, Apr. 1998.

[21] J. Kahan, M.-R. Koivunen, E. Prud'Hommeaux, and R. R. Swick. Annotea: An open RDF infrastructure for shared Web annotations. In *Proc. of the 10th WWW Conf.* [32].

[22] R. Killough and J. J. Leggett. Hypertext interchange with the Dexter model: Intermedia to KMS. TAMU-HRL 90-002, Hypertext Research Lab, Texas A&M University, Aug. 1990.

[23] O. Lassila and R. R. Swick (editors). Resource Description Framework (RDF) model and syntax specification. W3C Recommendation 22 February 1999, W3C, Feb. 1999. `http://www.w3.org/TR/REC-rdf-syntax/`.

[24] D. Martin and H. Ashman. Goate: Xlink and beyond. In Anderson et al. [2], pages 142–143.

[25] D. Michaelides, D. Millard, M. Weal, and D. D. Roure. Auld Leaky: A contextual open hypermedia link server. Position paper, OHS7, Aarhus, Denmark, 2001.

[26] D. E. Millard, L. Moreau, H. C. Davis, and S. Reich. Fohm: a fundamental open hypertext model for investigating interoperability between hypertext domains. In *Proc. of the 11th ACM Hypertext Conf.*, pages 93–102, San Antonio, TX, USA, May 2000.

[27] C. Nentwich, L. Capra, W. Emmerich, and A. Finkelstein. xlinkit: a consistency checking and smart link generation service. *ACM Transactions on Internet Technology (TOIT)*, 2(2):151–185, 2002.

[28] E. Sandvad, K. Grønbæk, L. Sloth, and J. L. Knudsen. A metro map metaphor for guided tours on the Web: the Webvise guided tour system. In *Proc. of the 10th WWW Conf.* [32], pages 326–333.

[29] K. Tochtermann, J. Westbomke, U. K. Wiil, and J. J. Leggett, editors. *Proc. of the 10th ACM Hypertext Conf.*, Darmstadt, Germany, Feb. 1999.

[30] R. H. Trigg. Guided tours and tabletops: tools for communicating in a hypertext environment. *ACM Transactions on Information Systems (TOIS)*, 6(4):398–414, 1988.

[31] F. Vitali, F. Folli, and C. Tasso. Two implementations of XPointer. In Anderson et al. [2], pages 145–146.

[32] W3C. *Proc. of the 10th WWW Conf.*, Hong Kong, May 2001.

[33] P. T. Zellweger, N. O. Bouvin, H. Jehøj, and J. D. Mackinlay. Fluid annotations in an open world. In *Proc. of the 12th ACM Hypertext Conf.*, pages 9–18, Århus, Denmark, Aug. 2001.