INFORMATIK

Berichte

295 – 8/2002

Proceedings of the International Workshop on Open Hypermedia Systems Core Concepts & Research Directions

Pre-Conference Workshop at
the ACM 13th International Conference on
Hypertext and Hypermedia (HT'02)
University of Maryland, College Park, MD 20742, USA
June 12th, 2002

David Millard, Jörg M. Haake, Sigi Reich (Eds.)

FernUniversität
Gesamthochschule in Hagen
FernUniversität
Fachbereich Informatik
Postfach 940
D-58084 Hagen

Proceedings of the International Workshop on Open Hypermedia Systems Core Concepts & Research Directions

Pre-Conference Workshop at
the ACM 13th International Conference on
Hypertext and Hypermedia (HT'02)
University of Maryland, College Park, MD 20742, USA
June 12th, 2002

David Millard, Jörg M. Haake, Sigi Reich (Eds.)

Organizers

David Millard Department of Electronics & Computer Science, University of Southampton, Highfield, Southampton, SO17 1BJ, UK, dem@ecs.soton.ac.uk

Jörg M. Haake FernUniversität Hagen Computer Science VI, Informatikzentrum, Universitätsstrasse 1, 58084 Hagen, Germany, joerg.haake@fernuni-hagen.de

Sigi Reich Salzburg Research (SunTREC), Jakob Haringer Str. 5/III, 5020 Salzburg, Austria, sreich@salzburgresearch.at

David Millard is a Research Fellow in the Intelligence, Agents and Multimedia Group at the University of Southampton. He has participated in the Open Hypermedia Workshops and Working Group meetings since 1997, where he was a key contributor to the OHP suite of protocols and the development of the Fundamental Open Hypermedia Model (FOHM).

Jörg Haake is professor for distributed systems at FernUniversität Hagen, the German distance learning university. He is actively participating in Hypertext conferences since 1991 and is engaged in OHS workshops since 1996, where his primary work is on collaborative open hypermedia systems.

Sigi Reich is the head of SunTREC (Sun Technology and Research Excellence Center) at Salzburg Research, Austria. He has participated in the Open Hypermedia Workshops and Working Group meetings since 1996. Over the last few years Sigi Reich has been involved in aspects of interoperability of Open Hypermedia Systems, in particular the development of the Open Hypermedia Navigational Interface (OHP-Nav) and the Fundamental Open Hypermedia Model (FOHM).

Table of Contents

Session 1: Linking and Structure

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 many 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], Webvise [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 Webvise [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 trough 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[1] 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`[2]: 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 contains a location specifier (locSpec) typical 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 the 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.

[1] http://www.daimi.au.dk/~les/ohif/ohif.dtd

[2] To distinguish OHIF elements from XLink elements their names are prefixed with a XML namespace.

`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.

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 references 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.

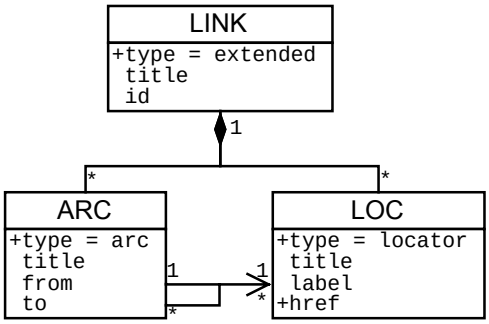


Figure 1: The structure of a xlink:link element. A xlink:link element can contain xlink:locs that each specifies a location with a XPointer extended URI. The xlink:arc elements connects the xlink:locs to form links.

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.

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(/,"UNIVERSITY OFAARHUS",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(/,"About the\
             Department",11,10)[1])"
 xlink:title="About the Department"/>

</LINK>
```

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.

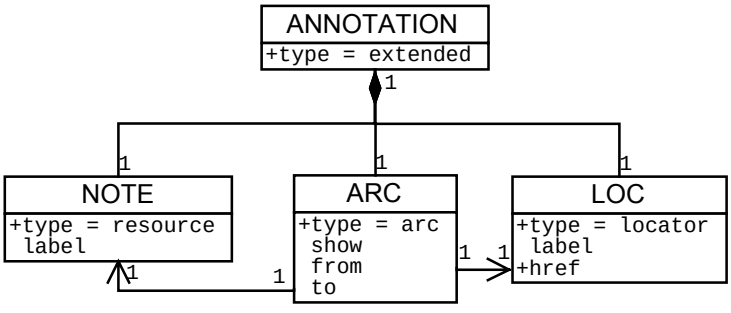


Figure 2: The structure of a xlink:annotation. The xlink:loc element specifies where the

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.

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`.

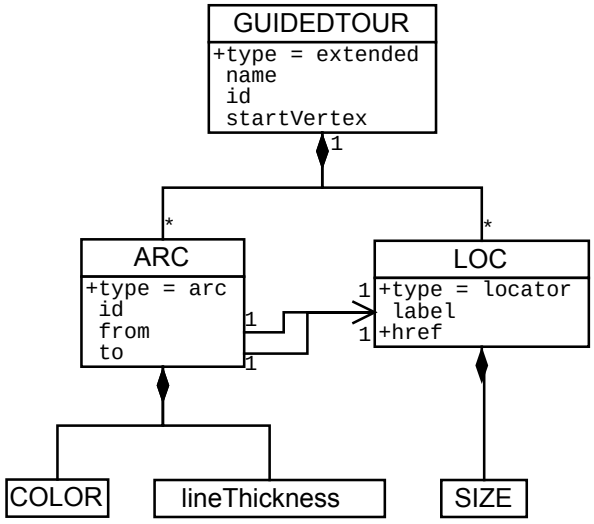


Figure 3: The structure of a xlink:guidedtour.

XSPECT—A SIMPLE IMPLEMENTATION OF XLINK

Our implementation, the Xspect system[3], 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.

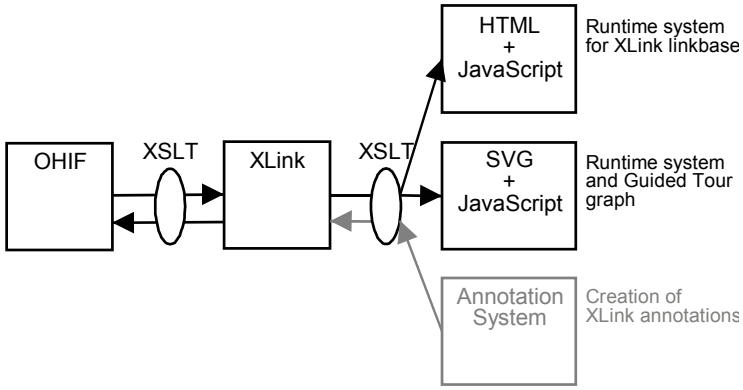


Figure 4: The Xspect systems transformation architecture.

[3] A demo of the system can be found on the URL:
`http://www.daimi.au.dk/~fah/xspect/start.html`

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.

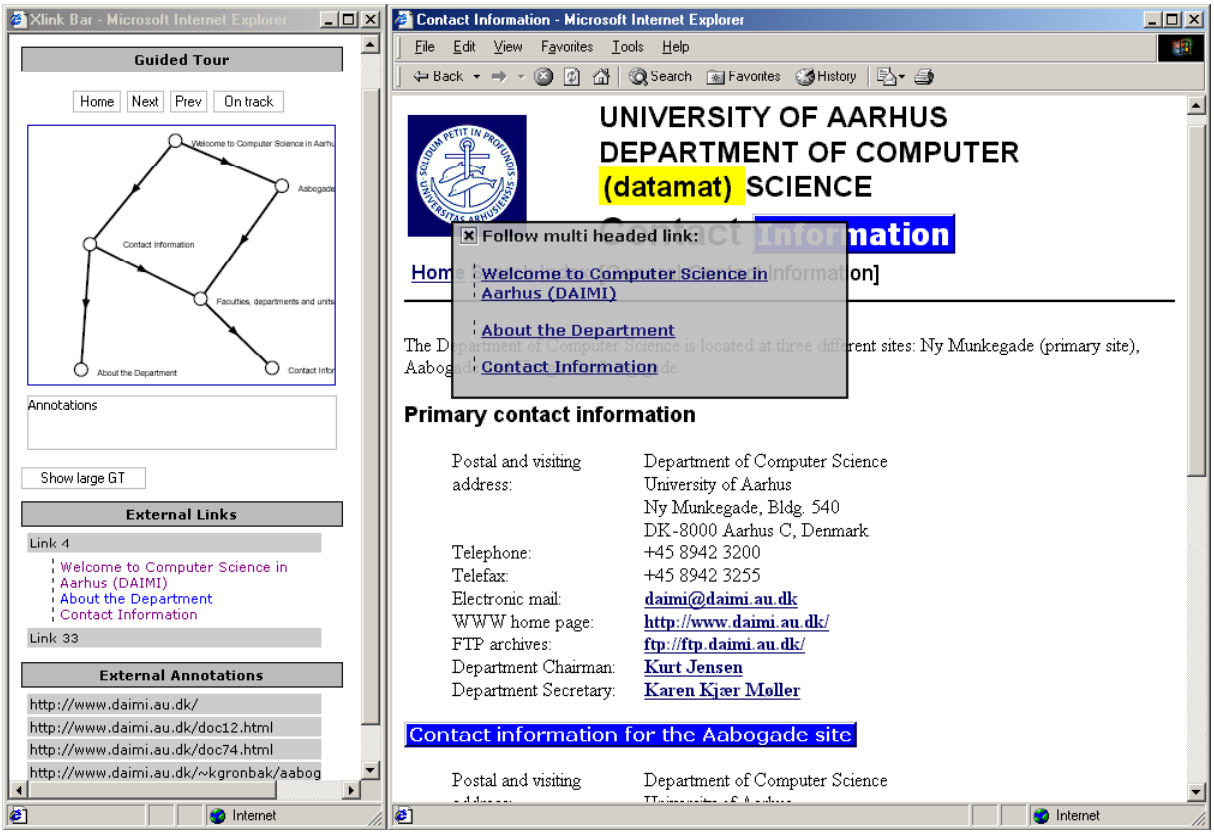


Figure 5: A session in the Xspect system. The left window displays a complete context with a Guided Tour, links, and annotations. The right window displays a document decorated with link- and annotation anchors. The user has activated a link and the corresponding link dialog is open.

DISCUSSION AND FUTURE WORK

We are now ready to discuss our expriences 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 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 whitin 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 WebDAV [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 advise and comments on improving earlier versions of this paper.

Bibliography

[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 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 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 Grønbæk. OHS interoperability—issues beyond the protocol. In Proceedings of OHS Workshop 4.0 held at Hypertext '98, Pittsburgh, June 20-24, 1998.

[9] Kaj Grønbæk, 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 Grønbæk, Lennert Sloth, and Niels Olof Bouvin. Open hypermedia as user controlled meta data for the Web. In Proceedings of the 9 International World Wide Web Conference, pages 553-566, Amsterdam, Holland, May 2000.

[11] Kaj Grønbæk, 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 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 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 ACM Hypertext Conference, Darmstadt, Germany, February 1999.

Goate: An infrastructure for new Web linking

Duncan Martin[1] & Helen Ashman[2]

[1]University of Nottingham
Jubilee Campus
Nottingham
+44 115 8451158
djm@cs.nott.ac.uk

[2]University of Nottingham
Jubilee Campus
Nottingham
+44 115 9514237
hla@cs.nott.ac.uk

Abstract

In this paper, we introduce a client-platform independent mechanism for implementing new linking standards.

The paper defines the terms low-level and high-level in relation to linking languages, and discusses how HTML, a low-level language, can be used as a basis for high-level linking.

We also describe *Goate*, a HTTP proxy that allows high-level linking to be used with ordinary HTML browsers, first taking a high-level overview of *Goate* and then discussing implementation details.

Introduction

The adoption of any new standard is dependant on the availability of compatible client software. Considering the Web specifically, adoption of a new standard requires the support of the authors of browser software, and even presuming this is forthcoming, as a public system there is no guarantee of the level of client software in use.

It is issues such as these that cause substantive 'lag' on new standard adoption and also preclude diversity of link specification methods in use.

This paper introduces a system to address these issues and allow the implementation of new linking languages for every browser.

The problem

If we want to take advantage of standards such as XLink[1], we have a problem since current Web browsers at best only have early support for XML. In terms of universal support we only have access to HTML, which lags XLink in terms of three key abilities; bi-directional links, *n*-ary links and flexible destination specification (i.e. the ability for the source link to specify where in the destination should be navigated to, XPointer[2] is an example of this).

However, XLink has the advantage of being a standard well supported by the computing industry and browser support is emerging. Now, suppose we look beyond XLink to linking based around ideas such as conceptual linking, a model that allowed links to be written in an arbitrary program language such as C or Java or other bespoke languages. Many of these future languages will not enjoy the wide industry support that XML & XLink has.

One solution to the browser manufacturers not supporting a new language could be to produce browser plug-ins. However, if we wanted our new linking language to be universally adopted we would have to produce (and maintain) a plug-in for every browsing platform in common use (where a browsing platform is a combination of Web browser and operating system) and frequently plug-in based approaches end up only supporting one or maybe two browsing platforms.

As an alternative to the plug-in approach, we intend to provide the three key abilities listed above whilst presenting only HTML (along with support languages such as CSS and JavaScript) to the browser, as part of a solution that allows the implementation of new linking languages whilst retaining full browser compatibility.

Solution theory

Our solution is based on the principle that all that is needed to support high-level linking is a capable display markup language and a low-level linking language.

Low-level linking

We define a 'low-level' linking as the ability to move from one page to another, the ability to specify the point in the destination page to which we wish to navigate and the ability to create links in the destination document. HTML has caveats in terms of the second and third of these abilities in that the in-page destination point must be pre-declared by the author of the destination document, and links can only be added to the destination document by the author of that document.

The term low-level is used to draw a parallel with programming languages. A low-level programming language (e.g. assembly) lacks many of the features of a high-level programming language (e.g. Java), yet it is capable of performing all the same tasks – albeit not with the same amount of effort from the programmer.

We complete the analogy by considering XLink (and future specifications) to be 'high-level' linking languages, as they allow link specification in a format tied closer to the concept desired.

If a linking language meets all of the criteria above it is a 'complete low-level linking language', whilst if it has shortcomings this will affect the range of languages that can be built upon it. For example, if the low-level language only allows in-page pointers to words rather than characters the high-level languages based upon it will also only allow in-page pointers to words.

Emulating high-level linking

We previously mentioned three key abilities present in XLink that are not in HTML; bi-directional links, *n*-ary links and flexible destination specification. If we treat these abilities as being a core requirement for a future (high-level) linking languages then they need to be modelled in HTML. The binary relationship linking model[8] shows that a uni-directional one-to-one link can be used as a basis for more complex linking, and we expand on this idea slightly with the following statements:

- A bi-directional link is equivalent to two uni-directional links that point at each other.
- *n*-ary links are equivalent to a collection of uni-directional links that share a common source point.
- Flexible destination specification can be emulated by placing a fixed in-page anchor at the desired point in the destination document, and navigating to this fixed anchor.

The key issues regarding this model are access and maintenance. We previously mentioned the caveats in considering HTML a low-level linking language restrictions which now becomes relevant since if we are to emulate flexible destination specification and back-links in HTML we need access to be able to write to the destination document. The maintenance issue is regarding bi-directional links, and ensuring that both uni-directional links remain 'in-sync' with each other.

The system we are implementing (named *Goate*[11]) handles these issues transparently, so that implementation of high-level languages remains in the high-level domain.

Working without access

The requirement to be able to write to the destination document isn't entirely accurate. A more precise requirement is: "the system needs to be able to alter the HTML as seen by the browser", and therefore writing to the copy 'on-disc' isn't necessary. Given that we can't expect to have write access to an arbitrary server on the Internet, it makes sense to intercept and alter the documents as they are delivered.

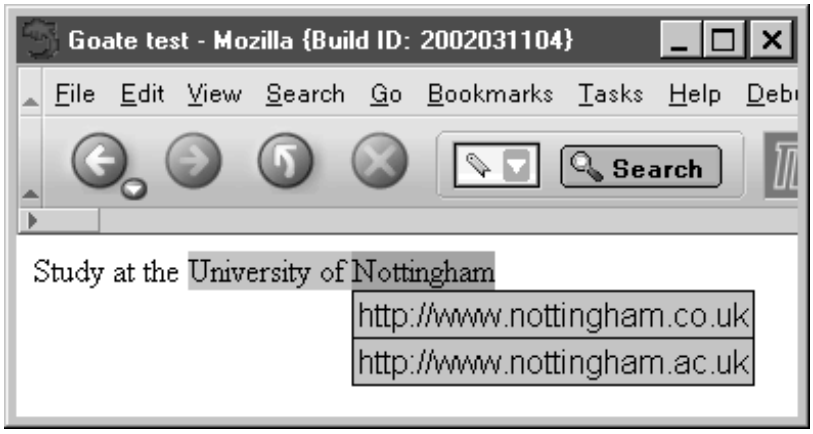
The dominant transmission protocol for documents on the Web is HTTP. There already exist HTTP proxies that act as relays for HTTP requests; that is clients send requests for pages to the proxy which then requests the document from the server. The reply from the server is sent to the proxy and from there to the client. Proxies are usually used for network infrastructure reasons (such as caching and controlled access to the Internet) and pass content verbatim, although it is possible for them to alter content as it passes through[5][7].

Systems based on this principle already exist, e.g. DLS[4][8] and Webvise[3] although they differ from *Goate* in that they aim to use the browser as one viewer in a larger hypertext system, the proxy being one method of adding links to documents. *Goate* is a purely proxy-only solution, focused on being high-level linking to the browser.

Presentation

Underlining as a visual cue for linking is not without problems, as discussed in "The look of the link"[9]. *Goate*, following recommendations made in "The look of the link" and uses background shading to identify links, with different colours distinguishing between single versus multi-headed links, and forward versus backward links. *Goate* uses light blue for a single forwards link, grey for a single backwards link and orange for any multi-headed link, values which are currently fixed although ultimately the user will be able to customise the visual appearance.

For multi-headed links a pop-up box is displayed when the mouse pointer passes over the link. The pop-up lists available destinations, again using background shading to show which are forward links and which are backwards. An example of the pop-up box with two forward links, is shown below:



For browsing-platforms that do not support the required Javascript and CSS to display the pop-up, a text-only representation is used instead.

Browser compatibility

Browser compatibility is one of the core aims of the project, since we do not believe the Web is best when restricted to a handful of browsing platforms. Working with HTML (and support) instead of plug-ins helps us in this regard as although implementations of JavaScript can vary between browsers, these differences are small when compared to those between two plug-in programming models.

Additionally, functioning as a proxy gives the system knowledge about which browsing-platform is being used as this information is sent as part of the User-Agent line in the HTTP request header, allowing the proxy to respond with HTML tweaked for that particular platform. The following browsers are fully supported by *Goate*: Internet Explorer 4+, Netscape 4.7 & 6.2+, Mozilla 0.99+, Opera 5+ and Konqueror 3+. We believe this set of browsers allows us to claim comprehensive cross-platform support, but more importantly allows the user to work with their preferred platform.

Implementation

This section goes into more depth about how the proxy functions, the stages of link translation and how language modules are used.

Basic structure

Goate itself is written in plain C (i.e. not C++) and is being developed under OpenBSD, although it is known to work under FreeBSD and we do not expect conversion to other UNIX variants to be difficult.

The proxy doesn't run as a single process, rather on startup *n* copies of the proxy automatically forked off. Each one of these child processes waits for incoming connections and handles them appropriately. Each child process connects to the *Goate* PostgreSQL[10] database which is used to store details about links, as will be explained in more detail later. Although PostgreSQL is used, no PosgreSQL specific features are as the demands of the database only extend as far as simple queries and inserts/deletes.

Document retrieval

When an incoming connection to *Goate* is made, the request is parsed and then passed on to the remote server (that is, the server holding the file requested). Non-HTML types returned are passed through directly, whilst HTML documents are processed as described below.

The browser name (e.g. Netscape) and version can be detected from the request and this information is stored for later use.

Parsing

The HTML document is parsed into an internal 'xmlDoc' format. A xmlDoc consists of a number of xmlItems where a xmlItem is either a section of plain text, a comment or an element (a further type; 'link' is explained later).

The parser takes cues from both SGML (HTML) and XML parsers. A presumption is made that the document should be valid XML (in terms of well-formedness) and so the parser supports self-closing elements, e.g. `<something />`. However, unlike most XML parsers, *Goate* is tolerant of mistakes such as stray angle brackets and attribute values missing quotes etc. Syntax mistakes such as these are corrected at parse time[4], whilst well-formedness mistakes are corrected at the next stage.

Well-formedness correction

Many linking languages rely on the document tree being well-formed, that is, every opening tag has a closing tag and tags are closed in the opposite order to which they are opened. *Goate* therefore corrects documents to be well-formed, adding and deleting tags as appropriate to make the document valid XML. The algorithm used corrects nesting errors (tags closed out of order), missing closing tags and extra closing tags.

Link translation

At this stage of the process, the document is now well-formed and the links embedded in it need to be translated into the internal *Goate* format.

The translation is done by the language modules available to the system. A language module is not part of *Goate* itself but a distinct piece of code linked into the system at run-time, similar to the way a browser plug-in is not part of the browser but interfaces with it, the principle of this is approach being that implementing linking languages should not be restricted to the authors of *Goate* but should be possible for any interested party.

Each language module in turn scans the xmlDoc looking for elements that it recognises as links. On finding one, the module is responsible for evaluating the link and translating it into the following form: destination page, destination start, destination end and directionality.

The start/end combination refers to the position of the link within the destination page. Note that this pair doesn't have to obey tree discipline so linking to 'white sheep' in:

```
Beware <b>fluffy white</b> sheep
```

[4] More accurately, some mistakes are corrected (angle brackets in text sections not using the `&gt;` notation for example) whilst others such as missing quotes around attribute values are ignored, since the quotes aren't stored as part of the xmlItem structure anyway.

with a single pair is valid even though a single closing element is caught within that range.

The 'directionality' attribute of the link simply refers to the link being uni-directional or bi-directional.

Once the language module has completed the evaluation of the link, it calls an API function with the details. In the case where the link has many destinations, the API call is made repeatedly. The element in the xmlDoc that contains this link has a flag set to show that a language module has successfully processed it.

When all language modules have completed their scan, *Goate* deletes elements shown as being successfully processed links.

Link insertion

When the language modules called the insert link method from the API, an entry was made in the link table of the *Goate* database[5].

Retrieving links that need to be displayed for this page is now simply a task of performing a SELECT on the database for the current page. This will retrieve not only links that have their source on the page, but the backwards part of bi-directional links sourced on other pages.

The links are inserted into the xmlDoc as 'link' type xmlItems. Since these items may be added in places that break the tree, the well-formedness corrector is run again to bring the tree back to a valid state.

Link rendering and transmission

The final stage of processing is to scan the document and convert the link-xmlItems to HTML code. The precise code output is dependent on the browser type detected during the request stage. Where links end up being nested a multi-headed link is rendered.

Whether single or multi-headed, the links here still rely on the HTML `<a href>` method. Therefore we cannot navigate to an arbitrary position in the destination document, only a point pre-declared with an `<a name>` anchor which will not exist for the links we're creating. We solve this problem by making the source link point to an in-page anchor we presume will exist (e.g. `href="somepage#goate123"`). When the user clicks on the link and *Goate* is processing 'somepage', it will add the destination anchor (named goate123 in our example) at the appropriate point as part of the link insertion procedure, allowing the browser to navigate to that point.

Acknowledgments

This work is supported by the EPSRC – Grant number 20164

References

[1] XLink specification. http://www.w3c.org/TR/xlink/

[2] XPointer specification. http://www.w3c.org/TR/xptr/

[5] In the case of a bi-directional link, two entries were added to the link table: one for each direction.

[3] Grønback, K., L. Sloth Ørbæk, P. Webvise: browser and proxy support for open hypermedia structuring mechanisms of the World Wide Web. Proceedings of the 8th International World Wide Web Conference. 1999.

[4] Carr, L.A., DeRoure, D., Hall, W. and Hill, G. The distributed link service: A tool for publishers, authors and readers. Proceedings of the 4th International World Wide Web Conference. 1995.

[5] Brooks, C., Mazer, M.S., Meeks, S. and Miller, J. Application-Specific Proxy Servers as HTTP Stream Transducers. Proceedings of the 4th International World Wide Web Conference. 1995.

[6] Ashman, H. Relation modelling sets of hypermedia links and navigation. Computer Journal 43. OUP 2000.

[7] Barrett, R. and Maglio, P.P. Intermediaries: new places for producing and manipulating Web content. Proceedings of the 7th International World Wide Web Conference. 1998.

[8] De Roure, D., El-Beltagy, S., Carr, L. and Hall, W. A Distributed Link Service using Query Routing. Poster session of the 8th International World Wide Web Conference. 1999. http://www.ecs.soton.ac.uk/~dder/qdls/

[9] Weinreich, H., Obendorf, H. and Lamersdorf, W. The look of the link – Concepts for the user interface of extended hyperlinks. Proceedings of ACM Hypertext 01. 2001

[10] http://www.postgresql.org

[11] Martin, D. and Ashman, H. Goate: XLink and beyond. Proceedings of ACM Hypertext 02. 2002.

Beyond the Traditional Domains of Hypermedia

David E. Millard, Danius T. Michaelides, David De Roure, Mark J. Weal
Department of Electronics and Computer Science
University of Southampton, UK

Abstract

The interoperability work of the OHSWG identified three major domains of hypermedia that needed to be addressed, Navigational, Spatial and Taxonomic. The Fundamental Open Hypermedia Model attempted to represent all three domains in one structural model and allowed context to be tackled consistently across the domains. In this paper we reflect on our experiences with creating contextual applications using FOHM and describe some of the structures that lie beyond the original three domains. We also explore some of the issues of having a generic model of context alongside a hypermedia model of structure.

1 Introduction

The Open Hypermedia Protocol (OHP) [2], developed by the Open Hypermedia Systems Working Group (OHSWG) was an attempt to define an interoperability protocol for Open Hypermedia Systems. As well as a communication protocol it also required a model of hypermedia that was acceptable to the larger community. To this end the OHSWG defined several 'domains' that would describe hypermedia's various application areas (Navigational, Spatial and Taxonomic Hypertext).

The original OHP definition became OHP-Nav, focused on navigational structures. It was envisaged that the remaining domains would be covered by other protocols (OHP-Space, OHP-Tax etc.)

Researchers at Southampton took the view that any model of hypermedia should encompass all three of these described domains [7], this would allow cross-domain browsing (such as following a Navigational Link to a Spatial Area) and also cross-domain fertilisation (where the facilities of one domain enhance another, such as Taxonomic branching becoming available with Navigational or Spatial structures).

This work resulted in the Fundamental Open Hypermedia Model (FOHM) [8] a generalised model of hypermedia capable of handling the three domains.

2 FOHM

FOHM is heavily based on the OHP-Nav model but it generalises that model in several ways:

- *Associations*. Rather than specify Links whose members must be either source, destination or bi-directional, FOHM specifies Associations, which contain a list of keys called a feature space. Each member of the Association must specify a value for each key, effectively binding itself to the Association at a particular feature vector. For example, in FOHM a Link is an Association with only one feature, called direction.
- *Context*. To support Taxonomic 'Perspective' structures, where a choice has to be made about what sub-categorisation is to be made, FOHM allows Context objects to be attached at various points of the structure. These are sets of key/value pairs that describe

in which context this part of the structure can be seen. Figure 1 shows context being used in a navigational link, in this case the context specifies that for an adult the link has two destinations but for a child it has only one. The Figure shows that when a child retrieves the link the context that specifies adults fails (shown in black) and the corresponding destination is pruned away (the grey structure is removed), therefore children only see a link with a single destination.

- *Behaviour*. In addition to Context objects, FOHM also allows Behaviour objects to be attached to any point of the hyperstructure. These are used by clients to record specific behaviour that might be required at certain events. For example, a behaviour attached to a document might contain instructions on how the client should modify the user's context given that they have read the document (e.g. to represent knowledge gained).

If we consider the set of all possible structure, shown in Figure 2, we can begin to appreciate the scope of FOHM. The OHP view was that the Navigational (Nav), Spatial (Space) and Taxonomic (Tax) structures were separate and that separate protocols would deal with each one.

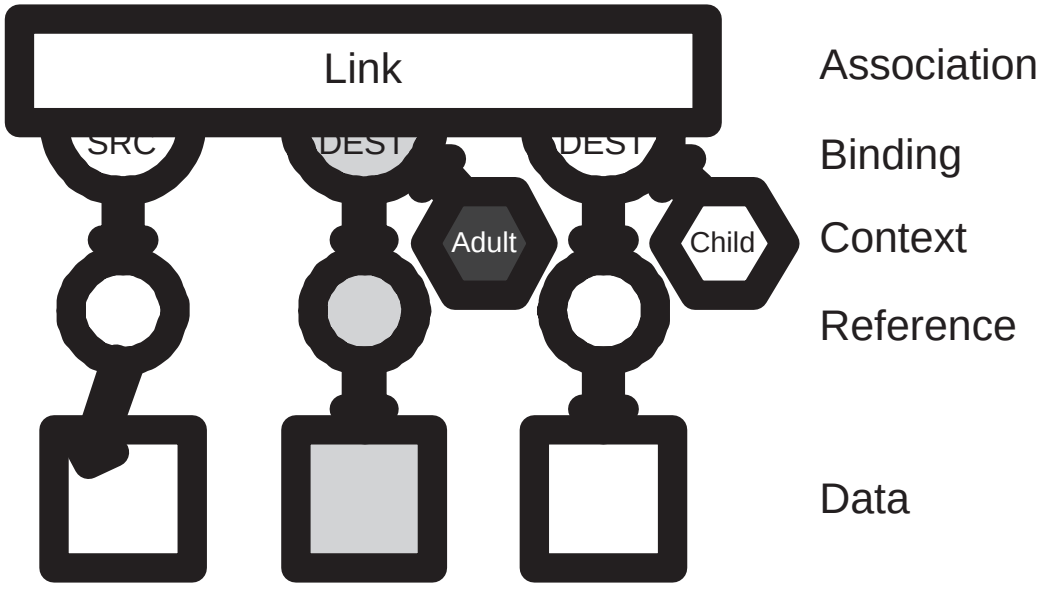
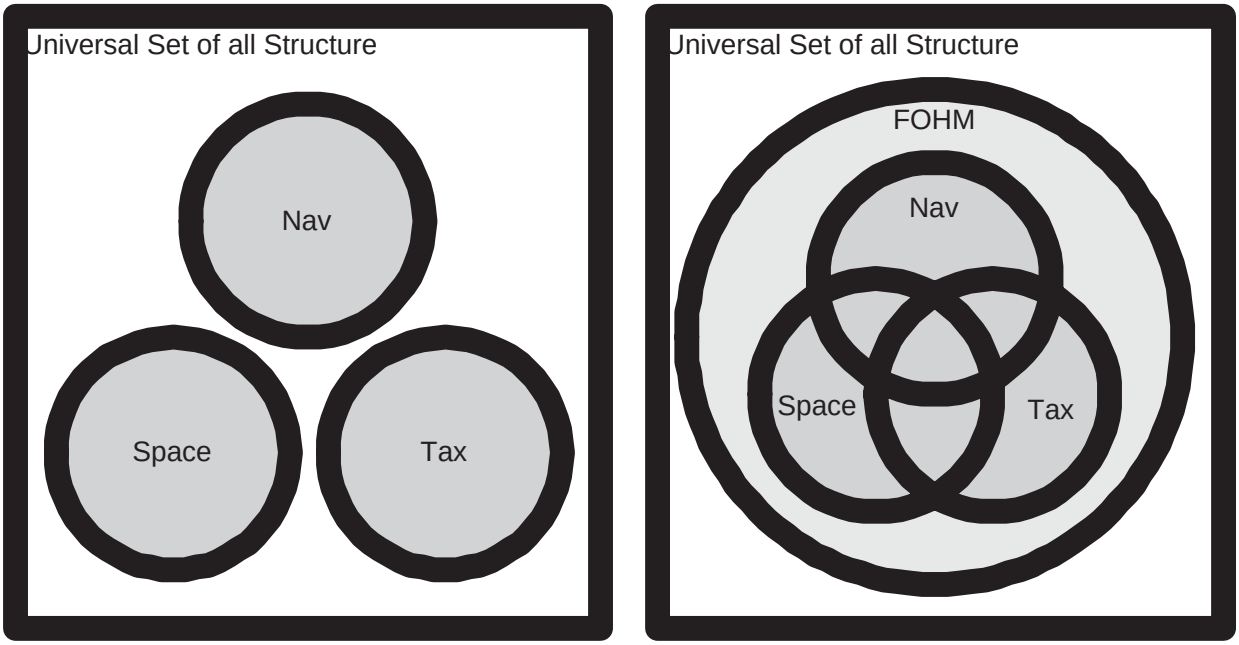


Figure 1: A Contextual Link in FOHM

The Southampton view was that the domains overlapped and that there were structures that would require a mixture of domains to express (such as a link to a space). FOHM was therefore an attempt to create a model capable of representing the union of all three domains (shown in dark grey in Figure 2).

However, FOHM is actually capable of expressing structure that lies outside of the three domains (shown in light grey). We do not make the claim that FOHM is capable of expressing all structure (i.e. that FOHM and the Universal Set are equal) although there is work that suggests that typed links, which may be represented in FOHM, are as expressive as generic metadata [10].



The OHP View of the OHP Domains The FOHM View of the OHP Domains

Figure 2: Beyond the Domains

3 Beyond the Domains

We have created a stand-alone Structure Server, called Auld Linky [6], that pattern matches FOHM structures and prunes them for context. In our work with Linky we have observed that many of the structures that we are using do not fall with the traditional domains but in fact lie outside of them (in the light grey area of the diagram).

In the following sections we will look at some examples of structure that do not wholly fit within a single domain. *Real-world Links* that extend the notion of a link beyond the OHP-Nav definition, *Virtual Documents* that extend Spatial structures with Taxonomic contexts, and *Sculptural Links*, a structure that, in OHP terms, exists in the overlap between the Navigational and Taxonomic domains.

3.1 Real-world Links

As part of the Equator project at Southampton we are investigating linking over real world spaces. In this case there is only one Data item, which represents the physical Universe. Areas in this Data item (areas within the real world) can then be referenced by name or location, just as words or paragraphs can be referenced in a normal hypermedia document. Links can subsequently be authored from one location to another.

In the Equator 'City' project we are applying Auld Linky and this real world linking metaphor to museum spaces. Visitors move around the museum with a hand held PDA and a positioning system (currently based on ultrasonics). When they move into a source area for a link the PDA shows them the suggested destination(s).

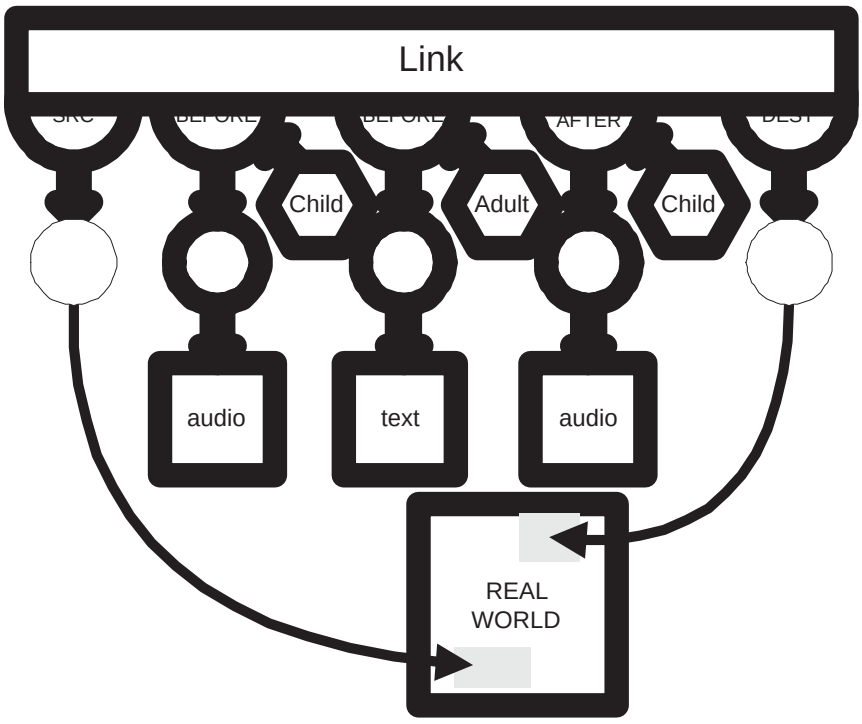


Figure 3: The City 'Real-world' Link Structure

One of the things we realised about the traditional source/destination based Link structure is that it does not allow for particularly complex information concerning the link itself. In the OHP-Nav model there is a *description* field, but this is just plain text and it is not clear if this is relevant before the link is followed, after it has been followed, or for overview purposes. One of the aspects of real world links is that, as visitors follow them by walking, the transition between source and destination can take some time. To cope with this and also to deal with our more sophisticated media requirements (we were particularly interested in using audio) we designed a new 'shape' to the traditional link, shown in Figure 3.

This new City link structure exists slightly beyond the original OHP domain of Navigational structure. It still has only a single feature called direction, but now items can be bound to it with any one of five values: 'Source', 'Destination' and 'Bi-Directional' as before but also as 'Before' or 'After'.

These last two values represent multimedia descriptions of the link that are appropriate to display to the user before and after they have made the link transition (i.e. an audio file bound as 'before' could be played as they walk towards the destination). These effectively offer the user *recommendations* for the next exhibit (Before) and *rationales* for the current exhibit (After).

The current system shows the user the destinations on a map. A possible extension to this is to add a sixth kind of binding that describes the 'directions' to the destinations, in effect informing the user of *how* to follow the link (something that in a digital link is hidden to the user but in a real-world environment needs to be expressed to them). This is made more complex because several different directions may have to be given to different destinations.

3.2 Virtual Documents

In another application we have experimented with virtual documents where a tour is constructed over many media fragments. Instead of following the tour one step at a time the client constructs

a document out of the tour members. Context objects attached to each member mean that membership of the tour is conditional and the document appears differently in different contexts.

This Xanalogical [11] idea is similar to conditional transclusion [9], where links are automatically resolved and their destinations may or may not be inserted into a document depending on the context. It is interesting to note that this is a contextual extension of a trail or tour structure (that have been described in hypermedia research for several years [5, 12]) but that even these basic tour structure do not fit into the three domains.

In terms of the original domains a virtual document is similar to a sequence of items from the Spatial domain and as it relies on context it also draws from the Taxonomic domain. However, virtual documents have different semantics to spatial list structures and thus belong beyond the spatial domain in the undefined area covered by FOHM.

3.3 Sculptural Hypertext

Recently the idea of Sculptural Hypertext has been suggested [1]. Sculptural Hypertext is distinct from traditional 'Calligraphic' node/link hypertext in that all nodes are initially interlinked. Meaningful hypertexts are thus constructed, not by making connections between nodes, but by removing them.

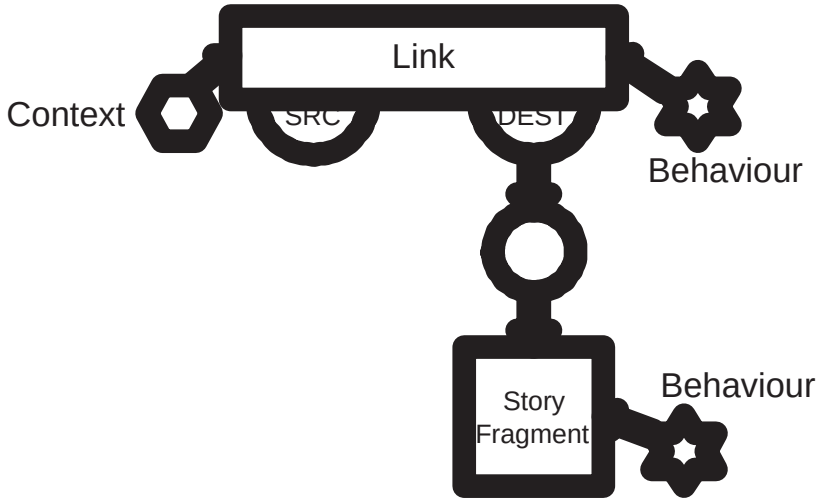


Figure 4: A Sculptural Hypertext Link

We have previously presented a sculptural hypertext system based on Auld Linky that uses link structures to facilitate sculptural hypertext [13]. The structure is shown in Figure 4.

It works by taking a generic link (as first described with Microcosm [3]) and generalising it even further. A generic link is one that has a source area that may appear in any document (for example glossary links). The sculptural link has a totally generic source that is valid in *all* positions of *all* documents. It works because the link has a context object attached to it and therefore only appears once the user's context matches with its own. As the user follows the sculptural links through the system, Behaviour objects on the link cause the users' context to evolve, this in turn reveals new links.

Because the sculptural links require context they could not be represented in OHP-Nav or the OHP-Nav model. In fact they exist in the intersection between Navigational and Taxonomic structure (as it is taxonomic hypertext that includes context).

4 Context vs. Structure

In the City project we have used hypermedia Anchors (References and LocSpecs in FOHM) to refer to actual locations in the real world. We have experimented both with named locations and also with regions (defined against map co-ordinates). We have then used context to model the user's preferences and knowledge (for example, previously visited exhibits).

However, it would also have been possible to use *context* to model the user's location, in effect scooping all the links according to context in the spirit of Sculptural Hypertext. A design decision had to be made about whether the user would search for links anchored in a certain region or whether they would search for all links but only in the context of the current region.

Early work on anchors described them as giving links context within a document [4], but as no other contextual dimensions were being considered, anchors were rapidly absolved into mainstream models. Only when we consider context fully can we appreciate that position is only one contextual dimension amongst many.

5 Conclusions

In our work with FOHM, Auld Linky and Context we have found a wealth of valuable structures beyond the original domains considered for OHP-Nav. Some of these, such as the contextual virtual documents, have been reminiscent of early hypermedia concepts, while others, such as the sculptural links, have helped to form new paradigms of hypermedia interaction.

These structures support the argument for developing models that deal with all hypermedia structure consistently, as opposed to specialising in a particular domain. All of these structures use or depend on context to add value. Our experience suggests that the border between what should be context and what should be structure sometimes becomes blurred. In particular, anchors, which specify position within a

larger whole, seem to be an aspect of context that history has caused to be treated specially.

It is not yet clear whether we should be making efforts to move other parts of hypermedia structure from the realm of context to that of defined structure, or whether we should be attempting to create a general model of context that we can use in place of specialised structure.

6 Acknowledgments

This research is funded in part by EPSRC IRC project "EQUATOR" GR/N15986/01. Thanks to the Equator City Project, particularly Matthew Chalmers and Ian McColl of the University of Glasgow.

References

[1] M. Bernstein. Card Shark and Thespis: exotic tools for hypertext narrative. In *Proceedings of the Twelth ACM Conference on Hypertext and Hypermedia, Arhus, Denmark*, pages 41–50, 2001.

[2] Hugh Davis, Siegfried Reich, and David Millard. A Proposal for a Common Navigational Hypertext Protocol. Technical report, Dept. of Electronics and Computer Science, 1997. Presented at 3.5 Open Hypermedia SystemWorking Group Meeting. Aarhus University, Denmark. September 8-11.

[3] Andrew M. Fountain, Wendy Hall, Ian Heath, and Hugh C. Davis. MICROCOSM: An Open Model for Hypermedia With Dynamic Linking. In A. Rizk, N. Streitz, and J. Andr´e, editors, *Hypertext: Concepts, Systems and Applications(Proceedings of ECHT'90)*, pages 298–311. Cambridge University Press, 1990.

[4] Linda Hardman, Dick C.A. Bulterman, and Guido van Russum. Links in Hypermedia: the Requirement for Context. In *Proceedings of the '93 ACM Conference on Hypertext, Nov. 14-18, 1993, Seattle, WA*, pages 183–191, 1993.

[5] Catherine C. Marshall and Peggy M. Irish. Guided Tours and On-Line Presentations: How Authors Make Existing Hypertext Intelligible for Readers. In *Proceedings of the '89 ACM Conference on Hypertext, Nov. 5-9, 1989, Pittsburgh, PA*, pages 15–26, 1989.

[6] Danius T. Michaelides, David E. Millard, Mark J. Weal, and David C. De Roure. Auld leaky: A contextual open hypermedia link server. In Siegfried Reich and Kenneth M. Anderson, editors, *OHS7 and SC3, Proceedings of the ..., To be published in Lecture Notes in Computer Science, Springer Verlag, Heidelberg (forthcoming)*, 2001.

[7] David Millard, Hugh Davis, and Luc Moreau. Standardizing Hypertext: Where Next for OHP? In Siegfried Reich and Kenneth M. Anderson, editors, *OHS6 and SC2, Proceedings of the ..., Published in Lecture Notes in Computer Science (LNCS 1903), Springer Verlag, Heidelberg (ISSN 0302-9743)*, pages 3–12.

[8] David E. Millard, Luc Moreau, Hugh C. Davis, and Siegfried Reich. FOHM: A Fundamental Open Hypertext Model for Investigating Interoperability Between Hypertext Domains. In *Proceedings of the '00 ACM Conference on Hypertext, May 30 - June 3, San Antonio, TX*, pages 93–102, 2000.

[9] Adam Moore, Timothy J. Brailsford, and Craig D. Stewart. Personally tailored teaching in WHURLE using conditional transclusion. In *Proceedings of the '01 ACM conference on Hypertext, Aarhus, Denmark*, pages 163–164, 2001.

[10] Graham Moore and Luc Moreau. From Metadata to Links. In Siegfried Reich and Kenneth M. Anderson, editors, *OHS6 and SC2, Proceedings of the ..., Published in Lecture Notes in Computer Science, (LNCS 1903), Springer Verlag, Heidelberg (ISSN 0302-9743)*, pages 77–86.

[11] Theodor Holm Nelson. *Literary Machines*. Published by the author. Mindful Press, 1987.

[12] Aggelos Pikrakis, Tilemahos Bitsikas, Stelios Sfakianakis, Mike Hatzopoulos, David C. De Roure, Wendy Hall, Siegfried Reich, Gary J. Hill, and Mark Stairmand. MEMOIR — Software Agents for Finding Similar Users by Trails. In *PAAM98. The Third International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents. March 23-25, London, UK*, pages 453–466, March 1998.

[13] Mark J. Weal, David E. Millard, Danius T. Michaelides, and David C. De Roure. Building Narrative Structures Using Context Based Linking. In *Proceedings of the '01 ACM conference on Hypertext, Aarhus, Denmark*, pages 37–38, 2001.

Asynchronous Linking in a Service-Oriented Architecture

Sanjay M. Vivekanandan, Kenneth Tso, Mark K. Thompson, David C. De Roure
Department of Electronics and Computer Science
University of Southampton, UK
{smv99r,kt00r,mkt}@ecs.soton.ac.uk

ABSTRACT

In this paper, we identify research issues in the development of system infrastructure support for asynchronous linkservices in a service-oriented architecture. We explore the suitability and applicability of using MQSeries Everyplace to provide a messaging backbone for linkservices that increases reliability, fault tolerance, and scalability. We identify and discuss some important problems and research issues related to this approach.

INTRODUCTION

We take the position that breaking the traditional synchronous nature of interactions between Open Hypermedia Systems components would engender reliability and scalability of services. We suggest that a service-oriented architecture, such as that offered by Web Services, readily enables hypermedia services to be published, deployed, and invoked by other services on both a global scale on the Internet, and also in a local-area peer-to-peer and pervasive scale. To enable asynchronicity between services, we suggest that store-and-forward middleware messaging systems, such as IBM's MQSeries Everyplace[7], provide the levels of communication decoupling required to meet this agenda.

This position paper introduces these concepts from this perspective and proposes an example implementation in a scenario where a user with a mobile device attempts to invoke linkservices whilst working in a disconnected state.

Distributed service-oriented architectures help create a distributed environment in which any number of services, regardless of physical location, can interoperate seamlessly in a platform– and language neutral manner. The success of any distributed service architecture is not only dependent on its ability to seamlessly integrate new and existing services, but also to function during periods of intermittent network connectivity.

In recent years, the Open Hypermedia Systems Working Group (OHSWG) has been working on a series of open hypermedia protocols to achieve interoperability between Open Hypermedia Systems[3]. The original Open Hypermedia Protocol (OHP)[6] effort was followed by the Fundamental Open Hypermedia Model (FOHM)[10], the latter concentrating on the link data model rather than an on-the-wire protocol. A contextual structure server, Auld Linky[9], has been developed grounded on the FOHM model and was designed to be a simple, lightweight structure server that serves according to contextual queries. The development of Auld Linky to date has not concerned security features or any level of transaction guarantee, for that has not been the focus of that group's activity to date. Recent work at Southampton has begun to investigate mechanisms for securing Auld Linky using MQSeries Everyplace[2].

MQSeries Everyplace (MQe) is designed to meet the needs of lightweight mobile devices such as phones and PDAs. It enables mobile devices to securely exchange messages both synchronously and asynchronously using queues and queue managers. Asynchronous messaging is vital in distributed architectures, for service providers and requestors cannot always depend on

the availability of each other to do their work. Through a system of queues, messages are exchanged in real time with transactional guarantee. During periods of network disconnection, messages are stored locally until a connection can be established and available for message delivery.

Service-Oriented Architecture

What is a Service-Oriented Architecture?

Service-oriented architectures (SOA) support a programming model that allows service components residing on a network to be published, discovered, and invoked by each other. Typically these services components interoperate with each other in a platform– and language independent manner.

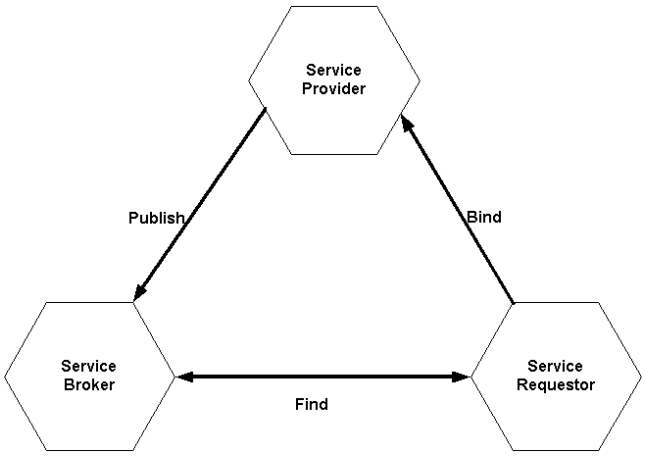


Figure 1. SOA, roles and operations.

The SOA consists of three core components: service brokers, service providers and service requesters (see Figure 1). A service broker acts as an intermediary between the service provider and the service requester, registers and categorizes published service providers and offers search services. A service provider deploys and publishes the availability of its services, and responds to requests to use its services. A service requester uses the service broker to find and bind to the desired service.

Web Services

The primary differences between a distributed service architecture and a distributed Web Service architecture is the size of the network being used and the underlying technologies involved. Web Services extend the SOA programming model into a vast networking platform that allows the publication, deployment, and discovery of service applications on Internet scale using Web technologies including SOAP[1] for inter-service communication, WSDL[4] for service description, UDDI[11] for service directories, and WSFL[5] for multi-service orchestration.

The Web Services platform is organized into the five layers of network, transport, packaging, description, and discovery, as described in Figure 2

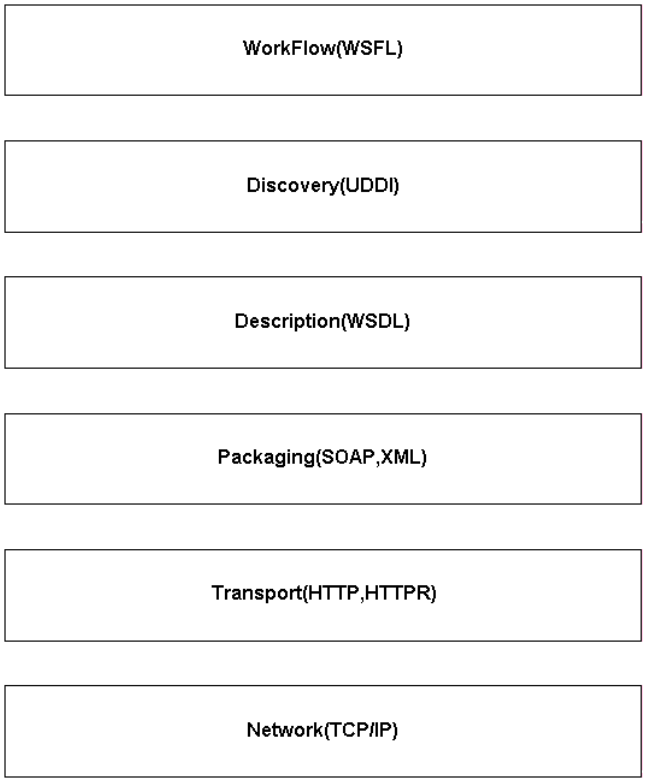


Figure 2. Web Services Technology Stack

Asynchronous Messaging in a SOA

Among the underlying requirements for SOAs to work effectively is that the network supporting the components and services need to be reliable, able to handle unpredictable loads, function during periods of intermittent network connectivity, and complete the ACID test for transactions. It can be argued that existing application connectivity models are neither sufficient nor necessarily appropriate for a pervasive computing infrastructure where participants in the architecture are not guaranteed to be available, discoverable or interact-able from moment to moment. To this end, we suggest architectures based on Message Oriented Middleware (MOM). MOM supports asynchronous messaging by using message queues as shown in Figure 3. Messages are exchanged between the service provider and service requestor through a system of queues. Messages from the service provider are sent to a queue, where the message stays until the service requestor is available and can read it from the queue.

From an OHS perspective, asynchronous service interaction readily enables selective and asynchronous link processing. When considering link resolution in a Distributed Link Service across multiple link services, where lock-stepped co-ordination between services is unlikely to be achieved, decoupling document content from resolved links is desirable. There are cases where the results of link resolution queries may no longer be required – perhaps the user is no longer reading the document – and thus the ability to propagate message cancellation on queues between application and services where the queries may not have yet been delivered to services, or the responses back to the client is desirable.

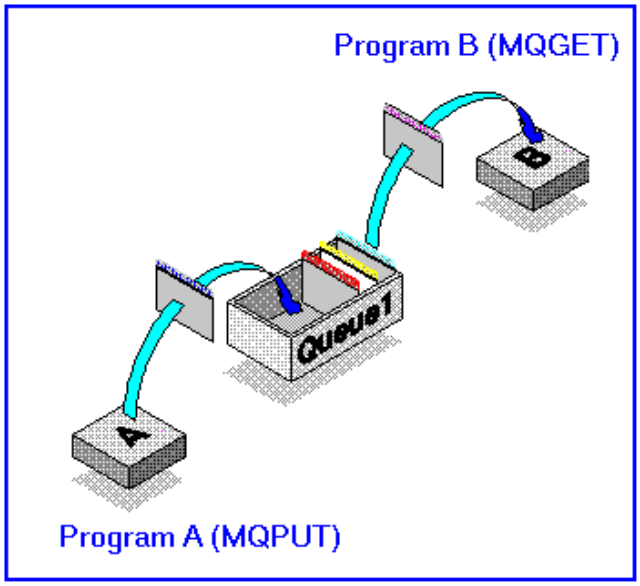


Figure 3. Message Oriented Middleware

MQSeries Everyplace Capabilities

MQSeries Everyplace (MQe) has a small execution footprint and can comfortably fit into modern mobile devices. In MQe, once-only message delivery is assured. Messages that are received for a remote device by MQe Queues will be temporarily held locally until it can be delivered to its final destination (i.e. when a connection is established). The length of time for messages to remain on the queue is defined by the queue expiry interval (e.g. 5 minutes). Once the time limited is exceeded, the message is marked expired and subsequent action (e.g. deleting it, move it to a dead-letter queue or re-sending it) is determined by a configurable rule in the queue manager process. Message Listeners can be added to the application to listen for events occurring on queues, such as message arrival. MQe provides many security features to protect the confidentially and integrity of messages as well as authenticating entities (e.g. queues, queue managers and users). Using MQSeries-bridge, messages can be exchanged with other MQSeries[8] family members, enabling integration of MQe-based services with pre-existing Enterprise applications.

Discussion

We propose the addition of MQe as a messaging backbone in a SOA that increases reliability, scalability, fault tolerance, and the loose coupling of providers and requestors. In Figure 4, Service A may wish to invoke the Leaky service but the user could be on a mobile device that holds an MQe queue. The user of the client device works offline and stores the SOAP call as a message in a MQe queue on the client device. During network access, the message is sent to a separate MQSeries input queue on the server. The MQSeries proxy retrieves the data from the MQSeries input queue, translating them to HTTP requests, and subsequently forwards it to the Leaky service. The response from the Leaky service is returned to the proxy, and places it in the output queue. MQSeries later sends the result of the query to the MQe queue on the client device, using a queue synchronization process.

Certain issues crop up with the usage of an asynchronous method of transport. The length of time a queue holds the message is among these issues. MQe queues can be defined with an expiry interval, and this function ensures that any message that has remained for a period longer then specified will be deleted. The type of service that is invoked is important in this aspect. If for example, a user queries an Auld Linky service that responds with a set of autobiographical links

of an author, the issue of message expiry is not paramount. However, it may be the case that the links returned are of critical importance requiring the client to be informed with haste. This raises the question on how and when to update the links, and how to manage the liveness of asynchronous queries between client and services. MQe does facilitate the concept of filtering which allows it to perform powerful search functions, thus allowing the client to receive messages with higher priority first (or with a shorter expiry time). One drawback to this method is that the links resolved would have to be pre-tagged before being send to the MQe queue.

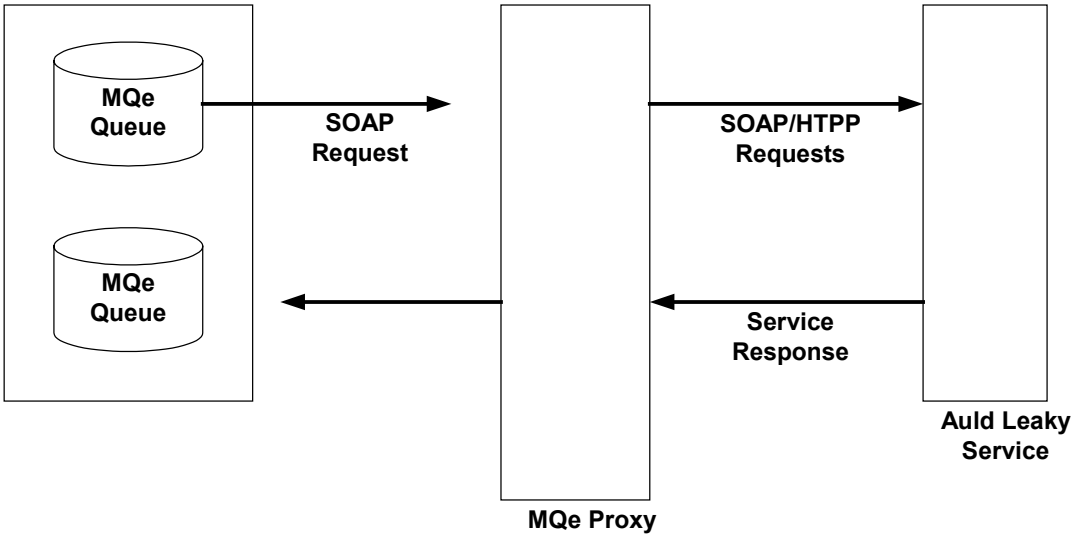


Figure 4. Invoking an Auld Leaky Service

In a peer-to-peer model, mobile devices may act like a service provider and service requestor simultaneously. Devices using MQe cannot exchange messages without knowing the target queue manager and queue names and hence cannot readily discover each other's services. One possible solution is to set up a server acting similar to a UDDI registry where service providers' queue manger and queue names together with the services they provide is stored and queried.

In this position paper we have begun to explore the addition of MQe as a messaging backbone that increases reliability in a SOA, focusing initially on application to link services. The goal of this infrastructure is to provide reliable asynchronous hypermedia services, and ensures that a transaction is completed once initiated.

To conclude, we suggest three areas of research relating to asynchronous linkservices in a SOA:

1) Asynchronous Link Processing

   For example, how should the query results be delivered to the user? Should it be in a separate window, and should it be loosely coupled with the user's interaction?

2) Message expiry

   For example, how long should the messages be stored in the queue before being deleted? How should the messages from different web services be handled?

3) Service Discovery

   For example, how can services provide by mobile devices in a peer-to-peer model be discovered and be invoked by each other?

REFERENCES

[1] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H.F. Nielsen, S. Thatte, D. Winer. Simple Object Access Protocol (SOAP) 1.1, W3C Note 08 May 2000. http://www.w3.org/TR/SOAP

[2] D.C Roure, K. Tso, H. Lambert. Securing a Open Hypermedia System (OHS) Using MQSeries Everyplace (MQe). Submitted to OHS2002, Maryland,USA.

[3] DAVIS, H. C.,MILLARD, D. E., REICH, S., BOUVIN, N.,GRØNBÆK, K., N¨URNBERG, P. J., SLOTH, L.,WIIL, U. K., AND ANDERSON, K. M. Interoperability between hypermedia systems: The standardisation work of the OHSWG. In Hypertext '99, The 10th ACM Conference on Hypertext and Hypermedia, Darmstadt, February 21-25,1999 (Feb. 1999), ACM, pp. 201–202.

[4] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana. Web Services Description Language (WSDL) 1.1, W3C Note 15 March 2001. http://www.w3.org/TR/wsdl.html

[5] F. Leymann. Web Services Flow Language (WSFL 1.0),IBM Software Group, May 2001. http://www-4.ibm.com/software/solutions/Webservices/pdf/WSFL.pdf

[6] Hugh Davis, Siegfried Reich, and David Millard. A proposal for a common navigational hypertext protocol. Technical report, Dept. of Electronics and Computer Science, 1997. Presented at 3.5 Open Hypermedia System Working Group Meeting. Aarhus University, Denmark. September 8-11.

[7] IBM MQSeries Everyplace, http://www-3.ibm.com/software/ts/mqseries/library/manualsa/manuals/mqsev12.html

[8] IBM MQSeries Family. http://www-4.ibm.com/software/ts/mqseries/

[9] MICHAELIDES, D. T., MILLARD, D. E.,WEAL, M. J., AND ROURE, D. C. D. Auld leaky: A contextual open hypermedia link server. S. Reich and K. M. Anderson, Eds.

[10] MILLARD, D. E.,MOREAU, L., DAVIS, H. C., AND REICH, S. FOHM: A Fundamental Open Hypertext Model for Investigating Interoperability Between Hypertext Domains. In Proceedings of the '00 ACM Conference on Hypertext, May 30 - June 3, San Antonio, TX (2000), pp. 93–102.

[11] Universal Description, Discovery and Integration, http://www.uddi.org

Session 2: Open Infrastructure

Securing a Open Hypermedia System (OHS) Using MQSeries Everyplace (MQe)

David C. De Roure[1], Kenneth K. K. Tso[1], Howard Lambert[2]

[1]Department of Electronics & Computer Science,
University of Southampton,
Highfield, Southampton,
SO17 1BJ, UK
{dder, kt00r}@ecs.soton.ac.uk

[2]IBM United Kingdom Ltd,
Hursley Park, Winchester
SO21 2JN, UK
howard_lambert@uk.ibm.com

Abstract

The Open Hypermedia Systems Working Group (OHSWG) has spent years working on Open Hypermedia Protocol and Open Hypermedia Systems. However, relatively less consideration is given to security, for instance, a contextual link server known as "Auld Leaky" was built with no security features at all. MQSeries Everyplace is designed with many security features necessary for building a secure open hypermedia system. "Auld Leaky" was chosen to integrate with MQSeries Everyplace making use of the security features. MQSeries Everyplace enables a secure client-server link service to be extended to a secure peer-to-peer distributed link service.

Introduction

In recent years, the Open Hypermedia Systems Working Group (OHSWG) has been continuously developing [3] and defining [8] the Open Hypermedia Protocol (OHP) [4] in an attempt to achieve interoperability between Open Hypermedia [1], [2], [5], [9], [10] Systems. Within the Intelligent Agents Multimedia group at Southampton, the Fundamental Open Hypermedia Model (FOHM) [7] based on the OHP model was developed. In addition, a contextual link server known as "Auld Leaky" [6] is constructed around FOHM.

Since "Auld Leaky" is a link server designed to be simple, lightweight but without considerations of security, it becomes susceptible to attacks over the Internet, in particular, when transmitting data in plaintext over an open network (e.g. via HTTP).

MQSeries Everyplace (trademark of International Business Machines Corporation) (MQe) provides sophisticated security capabilities (including authentication and encryption) to applications running outside the protection of firewall. By integrating with MQSeries Everyplace and making use of its security capabilities, "Auld Leaky" can be enhanced from a link server with no security to a link server with full security.

This paper describes firstly the security weaknesses of "Auld Leaky", and secondly, an overview of MQSeries Everyplace security features. Thirdly, one possible way of changing "Auld Leaky" into a secure Open Hypermedia System using the security features provided in MQSeries Everyplace is described. Finally, the potential of extending a secure client-server link service to a secure peer-to-peer distributed link service is also discussed.

Security weaknesses in "Auld Leaky"

The potential security threats or weaknesses of "Auld Leaky" are: (as security was not a prime concern when designing "Auld Leaky", several security weaknesses exist, which make it vulnerable to malicious users). First of all, transmitting data in plaintext over an open network via HTTP means that there is no confidentiality of the data. In addition, there is no authentication process and hence no control of access to "Auld Leaky" once its URL and port number is known. As a result, anyone could send all types of requests, including adding and/or deleting objects from the linkbases, and furthermore, no audit trail is kept showing the identity of the request or the type of requests to be processed.

An Overview of MQSeries Everyplace (MQe) security features

In the security world, there are four major areas: confidentiality, integrity, authentication and non-repudiation. Generally, in MQSeries Everyplace, confidentiality of message data is achieved by encryption. Different cryptors are provided, the choice is driven by the cryptographic strength needed to protect the data and complying with national security requirements. The use of SHA1 digest ensures the integrity of message data. Authenticators including NT authenticator, mini-certificates based on WTLS certificate are used for authentication purposes.

MQSeries Everyplace divides security features into four different categories known as local security, queue-based security, message-level security and link security to protect message data. Local security aims to provide protection for messages data held by a local queue manager using cryptors. Queue-base security concerns with protecting message data between an initiating queue manager and a target queue. Message-level security offers protection for message data exchanged between an initiating and receiving MQSeries Everyplace applications. Link security ensures the communication channels between queue managers are protected.

The architecture of a secure link service

Figure 1 illustrates how "Auld Leaky" can be integrated with MQSeries Everyplace. The client side simply contains a browser and a queue manager, which exists as an authenticatable entity. It begins with the browser collecting a query initiated by the user. A get message method is invoked to reliably send the query to the target custom queue on the server side. Then, the custom queue forwards the query to the link server and the link server response will be encapsulated in a message object before it is sent back to the client side.

From the security viewpoint, the architecture has several advantages; first of all, only the custom queue is allowed to directly communicate with "Auld Leaky" internally; secondly, access to custom queue is restricted to the server queue manager and subject to queue manager and queue rules. Together, this will provide access control to "Auld Leaky". Moreover, if four different custom queues are used instead of one custom queue to process the four different types of requests to "Auld Leaky", access control could be achieved at a more granular level. These requests are [6]:

- GET requests are sent with an ID (a simple string), the relevant object is returned in the form of XML if located in the linkbase.
- POST requests are sent with a FOHM object in the content of the message. This is then pattern-matched against the objects in the linkbase.
- PUT requests are sent with a FOHM object in the content of the message. "Auld Leaky" then adds this object to the linkbase.

- DELETE requests are sent with an ID, the relevant object is located in the linkbase and removed.

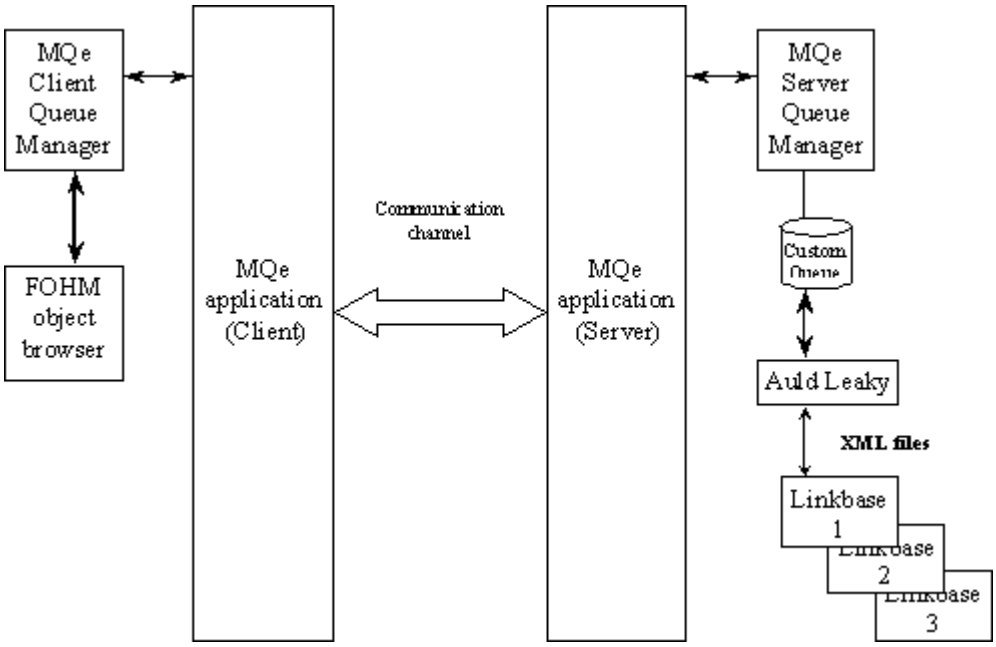


Figure 1. The architecture of a secure link server

The queries and responses exchanged over the insecure communication channel are protected by encryption and SHA1 digest. Queue managers having a digital signature and a mini-certificate become authenticatable entities. Furthermore, custom queues can keep audit trails in the form of messages.

Secure Distributed Link Service

The capabilities of MQSeries Everyplace is not limited by the security features described above. In addition, it is designed to support lightweight mobile devices. Particularly, its support of peer-to-peer connection enables the construction of peer-to-peer applications. Building a secure peer-to-peer distributed link service becomes possible by extending and modifying the secure link server to enable each peer to initiate and process multiple requests simultaneously. Figure 2 illustrates a concept of a secure distributed link service.

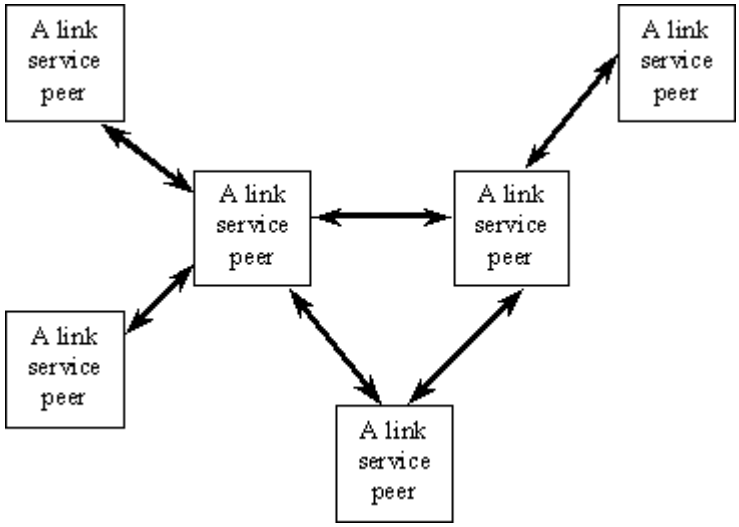


Figure 2. Secure peer-to-peer distributed link service

Conclusion

In this paper, the way MQSeries Everyplace could be used to secure "Auld Leaky" is described. This includes ensuring the confidentiality and integrity of queries and responses using encryption and SHA1 digest. Mini-certificates based on WTLS certificate owned by Queue managers on both client and server side are used to authenticate each other.

With the support of peer-to-peer connection, the secure client-server link service could be extended to a secure peer-to-peer distributed link service using MQSeries Everyplace.

Acknowledgements

This research is partially supported by department of Hursley Services & Technology at IBM United Kingdom. A special thanks is given to Dave Millard of Intelligence Agents Multimedia research group at Southampton for many helpful discussions.

References

[1] Anderson, K. M., Taylor, R. N. and Whitehead, E. J. Chimera: Hypertext for heterogeneous software environments. In *ECHT '94. Proceedings of the ACM European conference on Hypermedia technology, Sept. 18-23, 1994, Edinburgh, Scotland, UK (1994)*, pp. 94-197.

[2] Davis, H. C., Knight, S. and Hall, W. Light hypermedia link services: A study of third party application integration. In *ECHT '94. Proceedings of the ACM European conference on Hypermedia technology, Sept. 18-23, 1994, Edinburgh, Scotland, UK (1994)*, pp. 41-50.

[3] Davis, H. C., Millard, D. E., Reich, S., Bouvin, N., Grønbæk, K., Nürnberg, P. J., Sloth, L., Will, U. K. and Anderson, K. M. Interoperability between Hypermedia Systems: The Standardisation Work of the OHSWG. In *Hypertext '99, The 10th ACM Conference on Hypertext and Hypermedia, Darmstadt, February 21-25, 1999* (Feb. 1999), ACM, pp. 201-202.

[4] Davis, H., Reich, S. and Millard, D. A proposal for a common navigational hypertext protocol. Technical report, Dept. of Electronics and Computer Science, 1997. Presented at 3.5 Open Hypermedia System Working Group Meeting. Aarhus University, Denmark. September 8-11.

[5] Grønbæk, K. and Trigg, R. H. Design issues for a dexter-based hypermedia system. *Communications of the ACM 37*, 3 (Feb. 1994), 40-49.

[6] Michaelides, D. T., Millard, D. E., Weal, M. J. and DeRoure, D. C. (2001) Auld Leaky: A Contextual Open Hypermedia Link Server. In *Proceedings of the 7th Workshop on Open Hypermedia Systems, ACM Hypertext 2001 Conference. Aarhus, Denmark*.

[7] Millard, D. E., Moreau, L., Davis, H. and Reich, S. FOHM: A Fundamental Open Hypertext Model for Investigating Interoperability Between Hypertext Domains. In *Proceedings of the '00 ACM Conference on Hypertext, May 30 - June 3, 1992, San Antonio, TX, pages* 93-102, 2000.

[8] Reich, S., Millard, D. E. and Davis, H. C. (1999) Naming in OHP. *Proceedings of the 5th Workshop on Open Hypermedia Systems, ACM Hypertext '99 Conference, Darmstadt, Germany, February 21-25* p.43-47.

[9] Schnase, J. L., Legett, J. L., Hicks, D. L., Nürnberg, P. J. and Sánchez, J. A. Design and implementation of the HBI hyperbase management system. *Electronic Publishing-Origination Dissemination and Design 6*, 1 (June 1993), 35-63.

[10] Will, U. K. and Leffett, J. J. HyperForm: using extensibility to develop dynamic, open and distributed hypertext systems. In *ECHT '92. Proceedings of the ACM conference on Hypertext, November 30-December 4, 1992, Milan, Italy (1992)*, pp. 251-261.

Arguments for Open Structure Execution Service

Jessica Rubart[1], Weigang Wang[1], Jörg M. Haake[2]

[1]Fraunhofer Institute for Integrated Publication and Information Systems (IPSI)
Dolivostrasse 15
64293 Darmstadt, Germany
{rubart, wwang}@ipsi.fhg.de

[2]FernUniversität Hagen
Computer Science VI
Informatikzentrum, Universitätsstrasse 1
58084 Hagen, Germany
joerg.haake@fernuni-hagen.de

Abstract: The open hypermedia systems research community works on the provision of different standardized structure services in open environments. This position paper argues for the integration of services, which execute structure, to support different application domains. We describe two usage domains, in which structure plays a key role and where an open set of structure execution services is needed for the same structure. In addition, the same structure execution mechanisms are currently implemented redundantly in different systems rather than reusing structure execution services. Therefore, we propose a way to integrate those services using the concept of open service provision. Finally, a comparison is made between the current CB-OHS approach and the emerging Web services approach. This comparison from another angle reinforces the need for providing basic reusable business functions. It also highlights some weakness in our current OHS approach and suggests how we can strengthen it.

Keywords: component-based open hypermedia systems (CB-OHS), structural computing, cooperation support

1 Introduction

Open hypermedia systems research addresses interoperability and sharing of structure and behavior. The community is working on standardized interfaces to different structure services as well as on open system architectures to integrate and provide these services. This position paper argues for the integration of services that execute structure to support special application domains. The importance of separating behavior from structure and not only structure from data has been pointed out in [Nürnberg et al. 1996] in the context of hypermedia operating systems. From the structural computing point of view [Nürnberg et al. 1997] systems need an open set of behaviors, i.e. computations over structure. In this paper, we present our experiences with an open infrastructure that we use in the EU project EXTERNAL and talk about how open structure execution services can help to alleviate some of its problems. Then, we argue that object-oriented software development can be seen from a structural point of view and thus be supported by structure execution services, too. Finally, we propose a way to integrate those services using the concept of open service provision [Wiil et al. 2001].

2 EXTERNAL

The EXTERNAL project [EXTERNAL 2000-2002] aims at supporting Extended Enterprises, i.e. supporting the cooperation among (distributed) business partners. One important issue in this context is support for defining and executing work processes. The different partners of this project provide tools with complementary and partly overlapping functionality for this purpose.

In order to integrate the different tools we have specified a kind of open protocol for work process support based on a hypermedia data model. We have agreed on a special XML format, instances of which are used by the different tools and exchanged through a shared repository. The XML representations reflect a workflow structure. Thus, this format can be a basis for a standardized OHS-Workflow protocol.

We have recently adopted a service-centered approach regarding the integration of the different tools. **Figure 1** presents an abstract view on the service-centered infrastructure that we are currently using in the EXTERNAL project. End-users access the infrastructure through a common Web-based portal. The portal provides access to project-related information and enables modeling and execution of work processes (among other things). For this, the tools provide special APIs so that they can be invoked as different services on a specified workflow structure. The main modeling activities are done with METIS [Lillehagen and Karlsen 1999]. Workware [Jørgensen and Carlsen 1999] is used for enacting work processes, i.e. activate tasks and work on assigned documents. SimVision (formerly Vité) [Kuntz et al. 1998] provides probabilistic simulation of work processes facilitating, for instance, risk analysis. Finally, our component-based cooperative hypermedia systems XCHIPS [Rubart et al. 2001] is used for cooperative work management services, e.g. enterprise resource planning / management or joint modeling of shared work processes. The repository services provide support for persistence and access control of the XML instances. The different services can invoke themselves on specific workflow structure. **Figure 1** does not present explicit lines for all these possibilities. The infrastructure is open in the sense that new systems can be written or existing ones can be adapted to communicate with the existing services using the XML-based protocol.

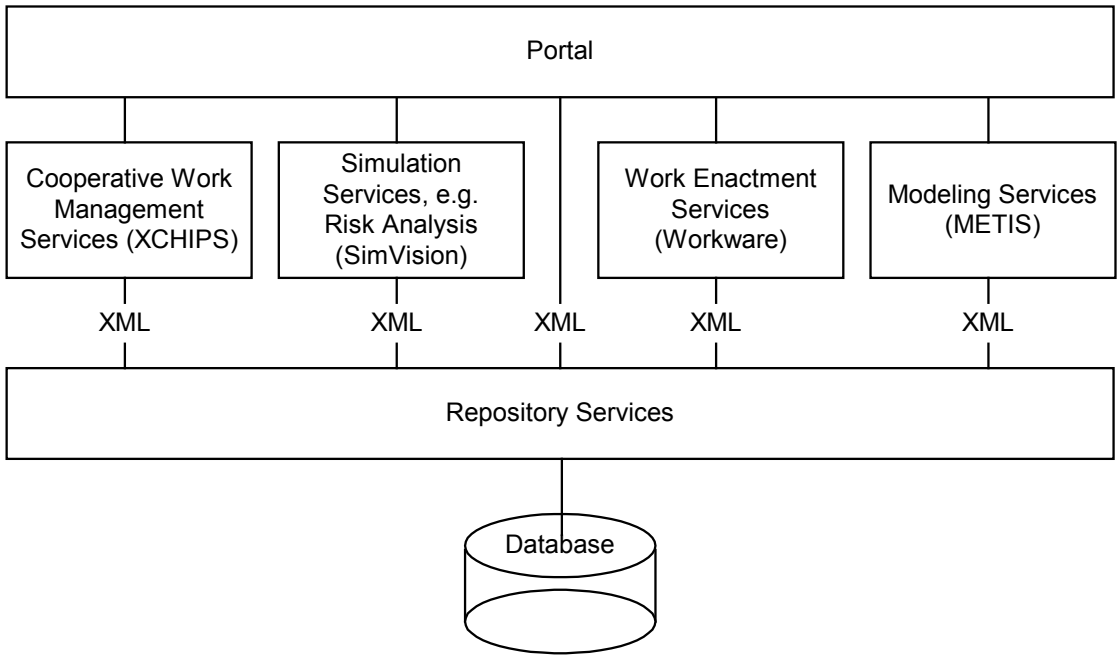


Figure 1. An Abstract View on the EXTERNAL Infrastructure

In the case of Workware and XCHIPS the APIs are implemented as extensions of Web servers, i.e. special URLs are provided for invoking the services on a given workflow structure represented using XML. Since the clients are based on Web technology (html and Java) only minimal installation effort is required.

In summary, the services in **Figure 1** work on the same structure. Speaking in terms of open hypermedia systems, the repository services incorporate foundation services and a kind of

structure service. The other services are clients using the repository services. Workware, SimVision and XCHIPS provide execution services on a workflow structure. In EXTERNAL, the number of execution services provided by them is growing. However, there is currently no abstraction for execution services so that an open set of them is supported. In addition, some functionality like, for instance, worklists are implemented in Workware as well as in XCHIPS. A separate worklist execution service could be (re)used by different systems. At this time, reuse of existing execution functionality of system A in system B is not possible.

Thus, we can conclude that an additional abstraction for execution services is very useful for open hypermedia systems since there might be different executions services needed for the same structure. The set of execution services needs to be open. In addition, some execution services need to be cooperation-aware, others not.

3 Object-oriented Software

For object-oriented software structure plays an important role. It deals with the composition of classes and objects and describes the ways in which classes or objects interact. The Unified Modeling Language (UML) [OMG 1998, Rumbaugh et al. 1999] is the standard notation for modeling object-oriented software. It is a visual modeling language that is used to specify, visualize, construct and document the static structure and dynamic behavior of object-oriented applications.

UML CASE (Computer Aided Software Engineering) tools like Rational Rose[6] or Together[7] support code generation on different UML models for several object-oriented programming languages as well as the creation of UML diagrams based on code. UML execution services could provide support for this and other tasks, such as generating documentation. And again, the set of execution services needs to be open since the same structure can be executed in several ways. In addition, these services could be (re)used by the different tools rather than implementing the functionality again.

4 Integration of Execution Services in the Concept of Open Service Provision

In [Wiil et al. 2001] the multiple open services approach is proposed. It includes a multi-layered architecture for the different types of services. Each layer is open to new services. **Figure 2** shows an execution services layer integrated in the concept of open service provision. It's a separate layer that is open to any number of execution services. The execution services (or basic business function services) use the (application-independent, generic) structure services. Any number of execution services can work on the same structure. Application A for example does not use any structure execution service, but is directly connected to a navigational structure service. Application B is connected to the metadata and workflow structure services and at the same time to an execution service for the workflow structure. Application C is connected to an object-oriented structure service as well as an execution service for code generation.

In [Tata et al. 2001] a cooperation services layer is proposed working on top of the structure services layer. Since the cooperation services layer is useful for the execution services as well, we propose to put the execution services layer on top of the cooperation services layer. This enables cooperation-aware execution services and non-cooperation-aware ones. In this context

[6] Rational Software, http://www.rational.com/products/rose

[7] TogetherSoft, http://www.togethersoft.com/

we can also think about moving the cooperation services layer into the foundation or structure services layers.

Existing open questions related to the distribution of the different services [Nürnberg and Leggett 1998] apply of course to the execution services as well, e.g. on which machines should the different services run?

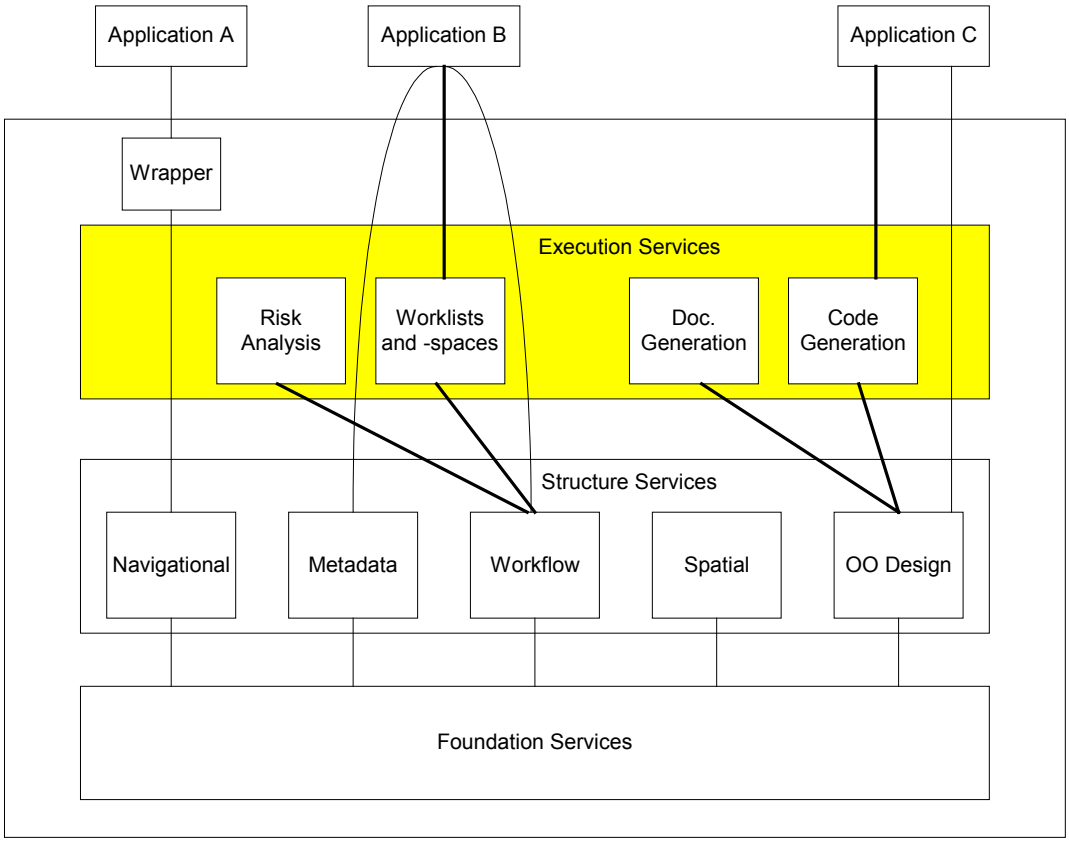


Figure 2. An Execution Services Layer integrated in the Concept of Open Service Provision

The separate abstraction of the execution services is valuable since we need several of them on the same structure. Putting all of them in the structure service is not open to new execution services. A new execution service then means that any implementation of the structure service needs to be updated. Putting them in the applications means that an execution has to be re-implemented in every application that needs it.

Regarding the integration in the user interfaces of the applications we might think about execution services that include a user interface and ones that do not include one. This implies that there might be a need for basic execution services without a user interface, which are used by execution services including a user interface. This means that horizontal interoperability for execution services working on the same structure is required.

Horizontal interoperability is generally important for composing structure and execution services.

5 OHS Services and Web Services

OHS services and Web services have a goal in common that is to make various systems open, linkable and interoperable. They both try to solve many of the same problems that were solved in

previous generations of computing: interface definition, endpoint routing, and data representation, however, they both do it in an Internet friendly way, i.e., through protocol interoperability and common transport.

The first OHS protocol, OHP-Nav, focuses on linking; while the newer OHS architectures, such as MOS and the above proposed extension of MOS, have broaden their scope to provide an open set of services (not only structural, but also functional) to various applications that confirm to certain open protocols. This is coincident to the approach of newly emerged Web service technologies. Although, they might have a short history, Web service technologies have developed very fast and become a quite effective and complete approach for enterprise integration. Therefore, for further development of OHS technology, it is very helpful to have a close look at Web service technologies and the Integrated Service Environments that make the development of Web services easier.

The Web service technologies provide a platform and vendor independent way to

- Make business functions readily shareable through WSDL (Web Services Description Language), SOAP (Simple Object Access Protocol) standards, and some newly developed Java XML and messaging APIs;
- Compose the shareable services into composite services or link them into a flexible process flow using various technologies, such as XLANG (Microsoft), or WSFL (Web Services Flow Language, IBM), or results of the WfMC (Workflow Management Coalition), or BPML (Business Process Modeling Language). The winner is yet to be decided;
- Deliver it in the right format (XML as the lingua franca, also Swing, Microsoft Foundation Classes or Wireless Markup Language); and,
- Make them discoverable and available to others anywhere (using UDDI standard – Universal Description, Discovery and Integration).

The typical Integrated Service Environments are the Microsoft VS.Net and some J2EE based Environments, such as SilverStream eXtend. VS.Net provides a single unified integrated development environment (IDE) for all languages to develop XML Web services and aggregate these services into applications. SilverStream eXtend provides Build & Orchestrate Services and Consume & Deliver Services to simplify the Web services development to a level that ordinary business application developers can easily do.

In the OHS community, there are also research efforts on service development environments [Wiil et al. 2001]. Comparing with Web service technology, OHPs function as a combination of WSDL and SOAP, but OHPs are instance protocols for different hypertext domains, rather than WSDL and SOAP kind of general mechanisms that can be used to describe and communicate with an open set of services. Similar to the common Web service result format, each hypertext domain has also a DTD describing its underlying structure. It should be possible to develop something similar to WSDL and SOAP, or to simply use them for accessing the OHS services. The purpose of the execution services proposed in this paper is to provide basic shareable business functions using the OHS technology (e.g., its structural services and foundation services). One of the basic business functions is to support flexible processes. Such a function can be developed as an OHS Workflow execution service upon the OHS Workflow structure services, so as to provide an alternative approach for services composition and for linking services into a flexible process flow. With the workflow execution services, we could support not only server-side composition but also the client-side composition (for applications) through the structure services and execution services.

6 Summary

In this paper, we proposed a new execution services layer for open hypermedia systems and showed how such execution services can be reused by different applications. Also, a comparison is made between the OHS execution services and Web services. This comparison has reinforced the need for a basic business function service layer that we identified in the enterprise modeling and operation application domain and the object-oriented software development domain. Through the comparison, we also identified the counterpart technologies and complementary technologies that we can also develop, learn from or simply make use of.

References

[EXTERNAL 2000-2002] EXTERNAL, EU Project, IST-1999-10091, New Methods of Work and Electronic Commerce, http://research.dnv.com/external/, 2000-2002.

[Jørgensen and Carlsen 1999] Jørgensen, H. D. and Carlsen, S. Emergent Workflow: Integrated Planning and Performance of Process Instances. In *Proceedings of Workflow Management '99*, 1999.

[Kuntz et al. 1998] Kuntz, J. C., Christiansen, T. R., Cohen, G. P., Jin, Y. and Levitt, R. E. The Virtual Design Team: A Computational Simulation Model of Project Organizations, In *Communications of the ACM*, vol. 41, no. 11, 1998, 84-92.

[Lillehagen and Karlsen 1999] Lillehagen, F., and Karlsen, D. Visual Extended Enterprise Engineering embedding Knowledge Management, Systems Engineering and Work Execution. In *Proceedings of IEMC '99*, IFIP International Enterprise Modeling Conference, 1999.

[Nürnberg et al. 1996] Nürnberg, P. J., Leggett, J. J., Schneider, E. R., and Schnase, J. L. Hypermedia Operating Systems: A New Paradigm for Computing. In *Proceedings of Hypertext '96,* ACM Press, 1996, 194-202.

[Nürnberg et al. 1997] Nürnberg, P.J., Leggett, J.J., and Schneider, E.R. As We Should Have Thought. In *Proceedings of Hypertext'97*, ACM Press, 1997, 96-101.

[Nürnberg and Leggett 1998] Nürnberg, P.J., and Leggett, J.J. Assessment of the Current State of Open Hypermedia Standardization Efforts. In *Proceedings of the 4th International Workshop on Open Hypermedia Systems* at Hypertext'98, 1998.

[OMG 1998] OMG: Unified Modeling Language Specification. Object Management Group, Framingham, Mass., Internet: http://www.omg.org, 1998.

[Rubart et al. 2001] Rubart, J., Haake, J.M., Tietze, D.A. and Wang, W. Organizing Shared Enterprise Workspaces Using Component-Based Cooperative Hypermedia. In *Proceedings of Hypertext'01*, ACM Press, 73-82, 2001.

[Rumbaugh et al. 1999] Rumbaugh, J., Jacobson, I., and Booch, G. The Unified Modeling Language Reference Manual. Addison-Wesley, 1999.

[Tata et al. 2001] Tata, S., Hicks, D.L., and Wiil, U.K. Cooperation Services in the Construct Structural Computing Environement. In *Proceedings of the 3rd International Workshop on Structural Computing* at Hypertext'01, LNCS, Springer Verlag, 2001.

[Wiil et al. 2001] Wiil, U.K., Hicks, D.L., and Nürnberg, P.J. Multiple Open Services: A New Approach to Service Provision in Open Hypermedia Systems. In *Proceedings of Hypertext'01*, ACM Press, 2001, 83-92.

Workflow Description for Open Hypermedia Systems

Sanjay M. Vivekanandan, David C. De Roure
Department of Electronics and Computer Science
University of Southampton, UK
{smv99r,dder}@ecs.soton.ac.uk

ABSTRACT

In this paper, we identify research issues in the development of system infrastructure support for introducing workflow support into Open Hypermedia Systems. We explore the suitability and applicability of having hypermedia services in a Web Services architecture, and integrating Web Services Flow Language for the coordination and interoperability of services. We identify and discuss some important problems and research issues related to this approach.

INTRODUCTION

We take the position that introducing workflow support into Open Hypermedia Systems (OHS) would enable coordination and integration of services. We suggest that a service-oriented architecture, such as that offered by Web Services, readily enables hypermedia services to be published, deployed, and invoked by other services on a global scale on the Internet. To enable integration and coordination between services, we suggest that workflow service components, such as IBM's MQSeries Workflow[7] and Web Services Flow Language (WSFL)[5], provide the levels of interoperability required to meet this agenda.

This position paper introduces these concepts from this perspective and identifies the research issues in the development of system infrastructure support for the composition of multiple OHS services.

Workflow deals with the management, specification, and execution of operations (*business processes*) in organizations. A business process is a coordinated set of work activities. It addresses the concerns of coordination of geographical and organizational distribution within distributed organizations.

Distributed service-oriented architectures help create a distributed environment in which any number of services, regardless of physical location, can interoperate seamlessly in a platform– and language neutral manner. The goal of the Web Services architecture is to simplify the development and integration of distributed services over the network, and one of the key aspects of this goal is to enable inter and intra enterprise business processes and workflows to seamlessly integrate new and existing services.

In recent years, the Open Hypermedia Systems Working Group (OHSWG) has been working on a series of open hypermedia protocols to achieve interoperability between Open Hypermedia Systems[3]. The original Open Hypermedia Protocol (OHP)[6] effort was followed by the Fundamental Open Hypermedia Model (FOHM)[10], the latter concentrating on the link data model rather than an on-the-wire protocol. In the OHS approach an open hypermedia system consists of three types of components: the client, a link or structure service, and a hyperbase or a linkbase. In the OHS architecture the interfaces between these components are clearly defined, and this allows each interface to be clearly defined as a Web Service. Any application or hypermedia system conforming to the respective interface definition can integrate with other OHS conformant systems. For example, any hypermedia system implementing the OHS client

interface can use OHS linking and navigation services provided by any OHS conformant link server. A Web Service architecture would allow these OHS services to be published, deployed, and invoked by other like- minded services on a global scale. WSFL builds on this scenario by building a framework in which service providers and consumers work together to implement and initiate standard business processes. This framework allows anyone who properly implements the appropriate OHS service interfaces to assume the various roles of OHS components.

Service-Oriented Architectures and Web Services

Service-oriented architectures (SOA) support a programming model that allows service components residing on a network to be published, discovered, and invoked by each other. Typically these services components interoperate with each other in a platform– and language independent manner.

The primary differences between a distributed service architecture and a distributed Web Service architecture is the size of the network being used and the underlying technologies involved. Web Services extend the SOA programming model into a vast networking platform that allows the publication, deployment, and discovery of service applications on Internet scale using Web technologies including SOAP[1] for inter-service communication, WSDL[4] for service description, UDDI[11] for service directories, and WSFL for multi-service orchestration.

The Web Services standard of primary interest in this paper is WSFL. WSFL is the Web Services Flow Language, and is an XML language for the description of Web Services compositions as part of a business process definition. It was developed by IBM to be part of the Web Services framework, and to complement existing standards and protocols. The WSFL specification considers two types of Web Services compositions:

- *Flow Model*: The Flow model describes how to choreograph the functionality provided by a collection of Web services to complete a particular transaction.
- *Global Model*: The Global model describes the interaction of a collection of Web Services with each other.

Workflow

Workflow Concepts

Workflow deals with the management, specification, and execution of operations (*business processes*) in organizations. Business processes are often automated using Workflow Management Systems (WfMS)[9]. WfMSs are tools that enable model-driven design, analysis, and simulation of business processes, which can be designed from scratch or from templates that support rapid application development. WfMSs also provide features for monitoring the execution of business processes and for automatically reacting to monitored events. In this section, we describe the basic workflow management system concepts that are used in the rest of this paper, and explain how these concepts are described in WSFL.

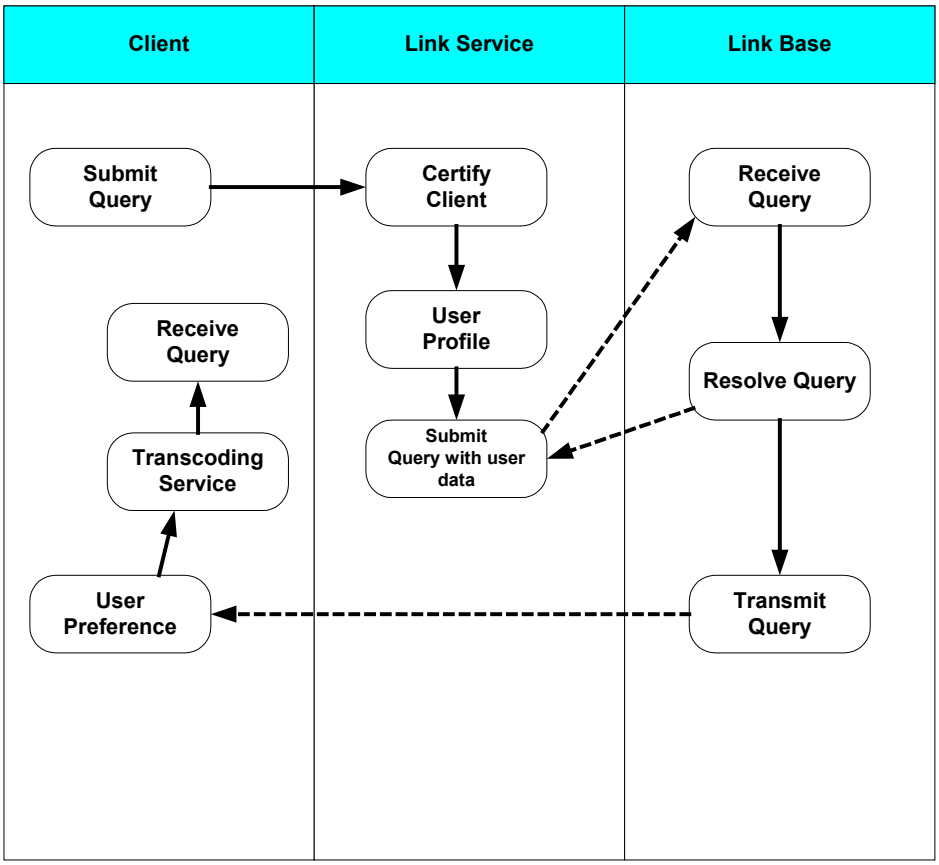


Figure 1. A Workflow Process

Workflow specifications usually describe the actions that are required to take place during the execution of business processes, and the overall flow of process. Figure 1 shows a workflow process modelled using a directed-edge graph. Each box is an activity (a transaction to be completed), and each activity is an individual Web Service, described by a WSDL document. All of the activities are linked together using arrows; called directed edges, that describes the flow of processing control from one activity to the next. Decisions are made at various control points to decide whether certain conditions have been met before the next activity is processed. The dotted-lines indicate the flow of information between activities. WSFL is essentially a tool to create an XML representation of the directed-edge graph that is both human and machine readable. By consuming WSFL, a workflow engine like IBM's MQSeries Workflow, can invoke and manage the entire business process.

Roles and Discovery

Every activity within a WSFL flow model is implemented in the form of a Web service offered by a Web service provider and represents the significant roles that must be filled to complete that process. Each service provider is expected to provide and implement the Web Service, or a composition of Web Services that would complete that transaction. From Figure 1, any Web Service provider that properly implements the Client, LinkService, and LinkBase Services may fill these roles. The fact that any OHS application adhering to the respective interface definition can interact with other OHS conformant systems allows service providers to fulfill these roles provided its compatible with the WSFL flow model for that process. The OHS Flow Model that corresponds with the graphical representation of Figure 1 is defined by the following WSDL specification:

```
<serviceProvider name =”Client”  type=”client”/>
```

<seriviceProvider name="Linkservice" type="linkservice"/>
<serviceProvider name="Linkbase" type="linkbase"/>

There are four different ways that the Web Services can be located: statically, locally, via UDDI, or dynamically while the transaction is being executed.

With a static location, the global model identifies a specific Web Service or composition of Web Services as the service provider for a given role. Local services are Web Services that are local to workflow engine processing the request. Locating a Web Service via UDDI essentially requires the global model to search the UDDI registry and retrieve a list of suitable Web Services. The global model decides on the Web Service by referencing a selection policy that may select the first service in the list, selecting a service at random from the list, or some user-defined algorithm. The use of UDDI allows multiple service providers to compete for the right to implement a role within a process. The ability to dynamically locate, and bind to service providers based on user defined selection policies adds a new dimension to conducting transactions on the Web that did not exist prior -- dynamic federation and integration of loosely coupled application components.

Recursive Composition

Recursive composition allows various service providers to combine services into a single solution. For example, a service provider may offer a LinkService Web Service that is actually a composition of Web Services provided from a number of different service providers (notification services, transcoding services, link resolver services). The end user only invokes the LinkService service, not the individual services that make up the LinkService service.

Discussion

In this paper we propose an approach to introduce workflow support for OHS systems. The goal of the infrastructure is to automate some of the services, making it simpler to maintain and integrate. We have also begun to explore the deployment of hypermedia services within a Web Services architecture, and integrating WSFL for the coordination and interoperability of these services.

To conclude, we suggest three areas of research relating to workflow support for OHS:

1. Application Interaction

   For example, how does a Web Service advertise its ability and willingness, to participate in a workflow process.

2. Reliability of Services

   For example, how do service providers guarantee the reliability of its services, and should there be a service-level agreement to guarantee reliability during a workflow process?

REFERENCES

[1] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H.F. Nielsen, S. Thatte, D. Winer. Simple Object Access Protocol (SOAP) 1.1, W3C Note 08 May 2000. http://www.w3.org/TR/SOAP

[2] D.C Roure, K. Tso, H. Lambert. Securing a Open Hypermedia System (OHS) Using MQSeries Everyplace (MQe). Submitted to OHS2002, Maryland,USA.

[3] DAVIS, H. C.,MILLARD, D. E., REICH, S., BOUVIN, N.,GRØNBÆK, K., N¨URNBERG, P. J., SLOTH, L.,WIIL, U. K., AND ANDERSON, K. M. Interoperability between hypermedia systems: The standardisation work of the OHSWG. In Hypertext '99, The 10th

ACM Conference on Hypertext and Hypermedia, Darmstadt, February 21-25,1999 (Feb. 1999), ACM, pp. 201–202.

[4] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana. Web Services Description Language (WSDL) 1.1, W3C Note 15 March 2001. http://www.w3.org/TR/wsdl.html

[5] F. Leymann. Web Services Flow Language (WSFL 1.0),IBM Software Group, May 2001. http://www-4.ibm.com/software/solutions/Webservices/pdf/WSFL.pdf

[6] Hugh Davis, Siegfried Reich, and David Millard. A proposal for a common navigational hypertext protocol. Technical report, Dept. of Electronics and Computer Science, 1997. Presented at 3.5 Open Hypermedia System Working Group Meeting. Aarhus University, Denmark. September 8-11.

[7] IBM: MQSeries Workflow: http://www-4.ibm.com/software/ts/mqseries/workflow/, 2002

[8] MILLARD, D. E.,MOREAU, L., DAVIS, H. C., AND REICH, S. FOHM: A Fundamental Open Hypertext Model for Investigating Interoperability Between Hypertext Domains. In Proceedings of the '00 ACM Conference on Hypertext, May 30 - June 3, San Antonio, TX (2000), pp. 93–102.

[9] The Workflow Management Coalition. The workflow reference model. Technical report, Workflow Management Coalition,http://www.wfmc.org/standards/docs/tc003v11.pdf.

[10] Universal Description, Discovery and Integration, http://www.uddi.org

Session 3: Lessons learned and Future Work

OHS: Lessons learned and Future Work

Jörg M. Haake[1], Dave E. Millard[2]
[1]FernUniversität Hagen
Computer Science VI
Informatikzentrum, Universitätsstrasse 1
58084 Hagen, Germany
joerg.haake@fernuni-hagen.de
[2]Department of Electronics and Computer Science
University of Southampton, UK
dem@ecs.soton.ac.uk

ABSTRACT

In this paper, we summarize the results of the closing session of the International Workshop on Open Hypermedia Systems held in conjunction with ACM Hypertext & Hypermedia 2002. The workshop participants identified three main areas of research in OHS: web technologies, hypermedia concepts, and open infrastructures. Lessons learned include: that the use of XLink is currently insufficiently specified, that HTML linking can be used to deliver higher-level linking services and that web services can be used as an infrastructure to deliver OHS concepts. The workshop participants felt that OHS technology was increasingly relevant to web research, as more and more web sites are beginning to manage their structure and content, and that more work is needed on making existing OHS knowledge accessible to the web community.

INTRODUCTION

This paper reports the results of the plenary discussion in the closing session of the International Workshop on Open Hypermedia Systems, which was held in conjunction with ACM Hypertext & Hypermedia 2002 on June 12, 2002. The workshop participants discussed three issues:

- current research themes in the field of OHS,
- lessons learned,
- future directions.

We briefly present the results of these discussions in the next three sections.

CURRENT RESEARCH THEMES

The participants identified three main areas of research in the OHS field: web technologies, hypermedia concepts, and open infrastructures.

Web Technologies

Several research groups reported about using Xlink as a syntax for providing OHS services [1, 3]. In addition, web services are currently evaluated as an infrastructure for the provision of OHS services on the web [4].

Hypermedia Concepts

Current topics on hypermedia concepts in the OHS community include

- the definition of low level linking languages, which can be used to deliver higher level linking services e.g. across the current Web infrastructure [2];
- research on the notion of context and structure in open hypermedia systems [1, 4] and
- the idea of asynchronous linking [6].

Open Infrastructures

Open infrastructure is a recurring issue in OHS meetings. Of particular concern are at this time the issues of security and open structure execution services:

- security in OHS includes authentification as well as access control and encryption of communication exchanges [2].
- Open structure execution services deals with the idea of having an OHS architecture that offers an extensible set of domain or tasks specific computations over structure [5].

LESSONS LEARNED

The participants agreed on the following lessons learned:

- The use of the XLink standard is by far under specified [1]. The OHS community can contribute to the field by supplying best practice examples of using XLink.
- Low level linking (e.g. by using HTML constructs) can be used to deliver higher level linking services [2].
- Web services could be used as an infrastructure to deliver OHS concepts and capabilities across the Web [6, 7, 5].

FUTURE WORK

More work is needed on Web infrastructure so that the lessons of more than a decade of OHS research can be applied to an increasingly managed Web, with the eventual objective of making the Web a platform offering advanced linking capabilities and OHS features.

The participants agreed that results of OHS research should be presented at WWW conferences. Best practice examples of using XLink are definitely needed in the Web community. New hypermedia concepts and examples of functioning OHS services and architectures should also be presented at Web meetings.

REFERENCES

[1] Bent Guldbjerg Christensen and Frank Allan Hansen. XLink—Linking the Web and Open Hypermedia. David Millard, Jörg M. Haake, Sigi Reich (Eds.), Proceedings of the International Workshop on Open Hypermedia Systems Core Concepts & Research Directions, Pre-Conference Workshop at the ACM 13th International Conference on Hypertext and Hypermedia (HT'02), (University of Maryland, College Park, MD 20742, USA, June 12th, 2002), pp. 9-18. Informatik Berichte, FernUniversität Hagen: Hagen.

[2] David C. De Roure, Kenneth K. K. Tso, Howard Lambert. Securing a Open Hypermedia System (OHS) Using MQSeries Everyplace (MQe). David Millard, Jörg M. Haake, Sigi Reich (Eds.), Proceedings of the International Workshop on Open Hypermedia Systems Core Concepts & Research Directions, Pre-Conference Workshop at the ACM 13th International Conference on Hypertext and Hypermedia (HT'02), (University of Maryland, College Park, MD 20742, USA, June 12th, 2002), pp. 40-44. Informatik Berichte, FernUniversität Hagen: Hagen.

[3] Duncan Martin and Helen Ashman. Goate: An infrastructure for new Web linking. David Millard, Jörg M. Haake, Sigi Reich (Eds.), Proceedings of the International Workshop on Open Hypermedia Systems Core Concepts & Research Directions, Pre-Conference Workshop at the ACM 13th International Conference on Hypertext and Hypermedia (HT'02), (University of Maryland, College Park, MD 20742, USA, June 12th, 2002), pp. 19-25. Informatik Berichte, FernUniversität Hagen: Hagen.

[4] David E. Millard, Danius T. Michaelides, David De Roure, Mark J. Weal. Beyond the Traditional Domains of Hypermedia. David Millard, Jörg M. Haake, Sigi Reich (Eds.), Proceedings of the International Workshop on Open Hypermedia Systems Core Concepts & Research Directions, Pre-Conference Workshop at the ACM 13th International Conference on Hypertext and Hypermedia (HT'02), (University of Maryland, College Park, MD 20742, USA, June 12th, 2002), pp. 26-32. Informatik Berichte, FernUniversität Hagen: Hagen.

[5] Jessica Rubart, Weigang Wang, Jörg M. Haake. Arguments for Open Structure Execution Service. David Millard, Jörg M. Haake, Sigi Reich (Eds.), Proceedings of the International Workshop on Open Hypermedia Systems Core Concepts & Research Directions, Pre-Conference Workshop at the ACM 13th International Conference on Hypertext and Hypermedia (HT'02), (University of Maryland, College Park, MD 20742, USA, June 12th, 2002), pp. 45-51. Informatik Berichte, FernUniversität Hagen: Hagen.

[6] Sanjay M. Vivekanandan, Kenneth K. K. Tso, Mark K. Thompson, David C. De Roure. Asynchronous Linking in a Service-Oriented Architecture. David Millard, Jörg M. Haake, Sigi Reich (Eds.), Proceedings of the International Workshop on Open Hypermedia Systems Core Concepts & Research Directions, Pre-Conference Workshop at the ACM 13th International Conference on Hypertext and Hypermedia (HT'02), (University of Maryland, College Park, MD 20742, USA, June 12th, 2002), pp. 33-38. Informatik Berichte, FernUniversität Hagen: Hagen.

[7] Sanjay M. Vivekanandan, David C. De Roure. Workflow Description for Open Hypermedia Systems. David Millard, Jörg M. Haake, Sigi Reich (Eds.), Proceedings of the International Workshop on Open Hypermedia Systems Core Concepts & Research Directions, Pre-Conference Workshop at the ACM 13th International Conference on Hypertext and Hypermedia (HT'02), (University of Maryland, College Park, MD 20742, USA, June 12th, 2002), pp. 52-56. Informatik Berichte, FernUniversität Hagen: Hagen.