Webvise: Browser and Proxy Support for Open Hypermedia Structuring Mechanisms on the WWW

Kaj Grønbæk, Lennert Sloth, & Peter Ørbæk
University of Aarhus
Department of Computer Science,
Åbogade 34, 8200 Århus N, Denmark.
Email: {kgronbak, les, poe}@daimi.au.dk

Abstract

This paper discusses how to augment the WWW with an open hypermedia service (Webvise) that provides structures such as contexts, links, annotations, and guided tours stored in hypermedia databases external to the Web pages. This includes the ability for users collaboratively to create links *from* parts of HTML Web pages they do not own and support for creating links to parts of Web pages without writing HTML target tags. The method for locating parts of Web pages can locate parts of pages across frame hierarchies and it is also supports certain repairs of links that breaks due to modified Web pages. Support for providing links to/from parts of non-HTML data, such as sound and movie will be possible via interfaces to plug-ins and Java based media players.

The hypermedia structures are stored in a hypermedia database, developed from the Devise Hypermedia framework, and the service is available on the Web via an ordinary URL. The best user interface for creating and manipulating the structures is currently provided for the Microsoft Internet Explorer 4.x browser through COM integration that utilize the Explorers DOM representation of Web-pages. But the structures can also be manipulated and used via special Java applets and a pure proxy server solution is provided for users who only need to browse the structures. A user can create and use the external structures as "transparency" layers on top of arbitrary Web pages, the user can switch between viewing pages with one or more layers (contexts) of structures or without any external structures imposed on them.

KEYWORDS: Open hypermedia, information structuring, navigation, collaboration, DOM

1 INTRODUCTION

The pioneers of hypermedia Bush [6], Nelson [23] and Engelbart [11] formulated hypermedia visions that included support for dynamic and distributed hypermedia structures which meant to include all human writings, and support people in searching, navigating, augmenting, commenting, structuring and collaborating in a giant "Docuverse". The World Wide Web [2] which appeared in the 90's has realized several aspects of these visions, and the WWW has become a popular and efficient means of distributing information with embedded hypermedia links on the Internet. The early visions put forward by the pioneers as well as later pre-Web systems, however, included more flexible support for people to create links, annotations and other structures between documents in the Docuverse. For example, the Intermedia[1] system

---

[1] Which by the way was the first system to use the term Web for a specific hypermedia structure.

[17] provided two-way links as objects between anchor objects stored apart from the document contents. With this approach Intermedia let users create two-way links between documents they did not own, as well as inspect which documents were linked *to* a given document. This idea of links and anchors as separate objects became the central idea of the Dexter Hypertext Reference Model [18] and systems based on this model, e.g. Devise Hypermedia [12],[16].

As of today, the WWW provides only little support for dynamic link creation and collaboration. But research on augmenting the WWW with link services that store link information separate from the document contents is underway. HyperWave [21], Microcosm's Distributed Link Service (DLS) [7], DHM/WWW [15] and Chimera [1] are examples of systems that support non-embedded links to WWW pages on the Internet and link storage in hypermedia databases on Internet servers.

These next generation WWW systems are, however, only in their infancy with respect to providing: linking inside pages, in particular non-text pages; linking in documents as they are being edited; collaboration support and distribution of hypermedia databases. This paper contributes to the design of next generation Web systems by discussing an approach to augmenting the popular Web browsers with support for external hypermedia structures. The approach is based on the work by Grønbæk and Trigg [14] on extending the Dexter model to handle both link objects and embedded addresses. The work is based on the ideas of the first DHM/WWW prototype built on top of the Devise Hypermedia (DHM) framework [15]. The prototype system described in this paper is called Webvise and is developed within the COCONUT project and thus available from http://www.cit.dk/coconut. With respect to choice of Internet browser, the Webvise Client is currently only integrated with the Microsoft Internet Explorer 4.x for Windows 95/NT. The reason for this is that the Microsoft Internet Explorer, to the best of our knowledge, is the only wide spread internet browser that is sufficiently open to support the kind of integration we aim at. Especially it is the exposure, through COM interfaces, of the Dynamic HTML's DOM [22] supported by the Microsoft Internet Explorer that makes the smooth integration possible.

The structure of the paper is as follows: Section 2 gives a brief introduction to the notion of open hypermedia and hypermedia structures external to document contents. Section 3 describes the user interface of the Webvise Client, and its extensions to MS Internet Explorer. Section 4 discusses the architecture of the Webvise Client and server components. Section 5 discusses central issues about accessing and managing external structures, including link integrity issues. Section 6 discusses different types of use of Webvise through use scenarios for the support provided by Webvise. Section 7 concludes the paper and points to some ongoing and future work in this area.

2 OPEN HYPERMEDIA SYSTEMS AND STRUCTURES

This section introduces the notion of open hypermedia and structures managed externally to document contents. The WWW use embedded unidirectional links also known as embedded addresses whereas open hypermedia system designers provide n'ary bi-directional external link objects stored in separate databases [12], [9], [7], [14], see Figure 1. We claim that both of these types of structures are useful and necessary in order to support the kind of dynamic hypermedia discussed in Section 1.
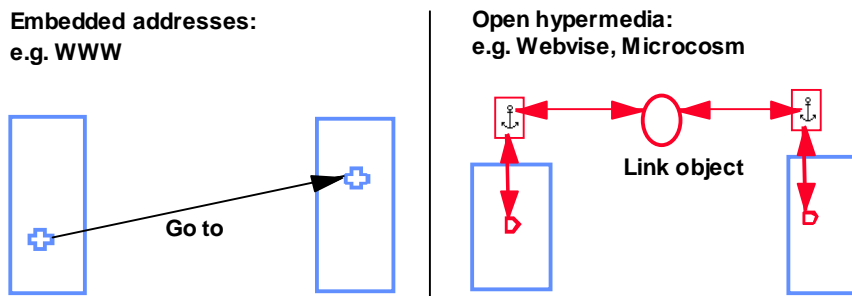
Figure 1: Embedded addresses and open hypermedia with external links

The biggest advantage of the embedded links in WWW is their simplicity: there is no need for a specialized link server, and the WWW only has to manipulate tagged ASCII files. This simplicity comes at a cost however, as only the owner of a document can create links from the document. At the same time links to specific parts of a document can only be made if there are already target tags at the desired point in the document. It is impossible to see which documents point to a document, and there can only be one set of links from a given document. If two users wish to have different links from the same document, they must maintain two copies of the document, identical apart from the different links. This requires extra maintenance, if the original document is later changed.

This situation is further complicated if the original document is not a simple ASCII file. Documents must first be converted into HTML before they can be properly used in a WWW context. This problem has been addressed by a variety of conversion programs, from simple RTF to HTML conversion to large-scale WWW publishing systems. Powerful as some of these systems are, their existence is indicative of the limited nature of HTML. Finally, links as implemented in the WWW are unidirectional and with only modest typing support.

External links are considerably more complicated, but offer advantages over embedded links. Because the original document is unaffected by link creation, users can create, maintain and share their own links for a given document, regardless of who owns the document. As links in systems based on the Dexter model [18] are first class objects in a database, links to and from a document can be traced from the document by querying the link database. Links may be named and typed, so the user can differentiate between e.g. a quote link and reference link. Typed links have been found to be of considerable value in many hypermedia systems [31], [22]. Unlike the WWW, external links can have more than one target. Finally as linking is handled outside the documents, the documents can be of any format.

Anchors act as reference objects for links and are responsible for encapsulating location information for a piece of information inside a document. Locating via anchors is not tied to text, and is limited only by the methods applications provide for accessing parts of their data. Anchors can be located in segments of sound and video, areas of pictures, or rows in a relational database.

2.1   Open hypermedia structures

This section gives an overview of the external structures that are provided by open hypermedia (http://www.ohswg.org). Similar mechanisms are proposed in Dexter model extensions made by Grønbæk and Trigg [13], in the OHP 2.0 proposal [8] as well as various open hypermedia systems [1], [19], [15], [26], [32]. Based on these proposals the open hypermedia community is implementing a common data model as depicted in Figure 2.
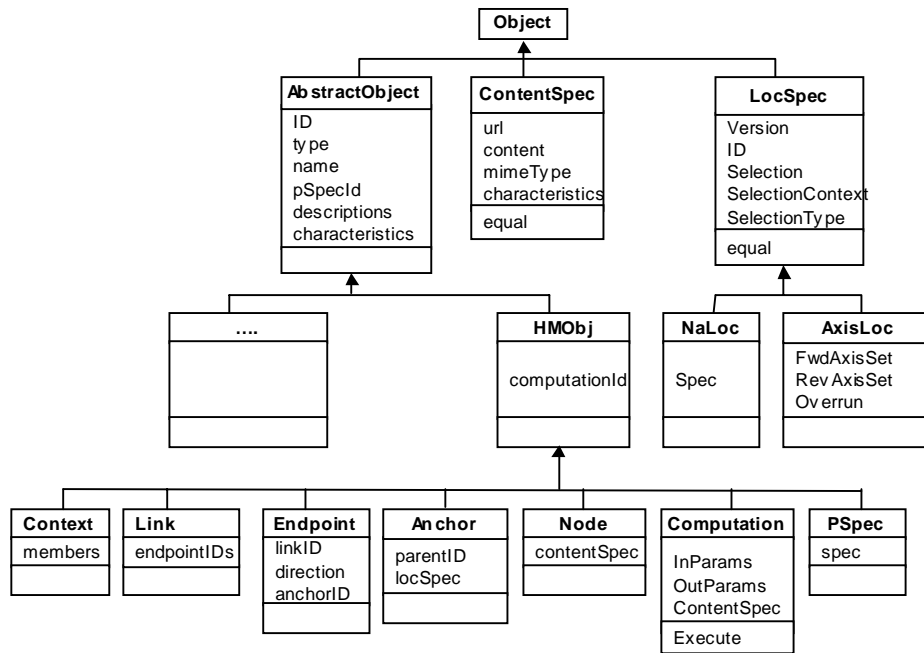
Object

AbstractObject
- ID
- type
- name
- pSpecId
- descriptions
- characteristics

ContentSpec
- url
- content
- mimeType
- characteristics
- equal

LocSpec
- Version
- ID
- Selection
- SelectionContext
- SelectionType
- equal

....

HMObj
- computationId

NaLoc
- Spec

AxisLoc
- FwdAxisSet
- RevAxisSet
- Overrun

Context
- members

Link
- endpointIDs

Endpoint
- linkID
- direction
- anchorID

Anchor
- parentID
- locSpec

Node
- contentSpec

Computation
- InParams
- OutParams
- ContentSpec
- Execute

PSpec
- spec

Figure 2: OHP — core navigational data model — generalization.

The specialization hierarchy in Figure 2 has general classes holding shared properties such as identifier, name etc. The first class hypermedia objects (context, link, endpoint, anchor, node, etc.) that contain unique identifiers inherits from these classes as well as the specifiers used to communicate content location (contentSpec) and anchor locations (locSpecs). These objects are typically associated to each other through one-to-one, one-to-many, and many-to-many relationships as described in further detail in [27].

Anchors contain a parentId, which is a HMObject id; this allows the generality, in which both a Link and a Context may be the parent of an Anchor where the LocSpec may locate, e.g., an endpoint in a Link or a member in a Context. Context is a class of objects similar to Hypertexts in Dexter [18]. These HMObject subclasses also include two classes used for specifying behavior at runtime: PSpec and Computation. PSpec is a specification that typically stores an opaque specification of application specific attributes that govern the presentation of an object at runtime. Computation is a class of objects which stores code in a programming language to be executed at runtime by an interpreter or virtual machine.

The specialization hierarchy in Figure 2 includes a number of specification classes: ContentSpec and LocSpec, and two LocSpec subclasses, which are used to instantiate specification objects that typically become in-lined in Node or Anchor objects. Thus, these objects contain no global unique identifier. ContentSpec is used either to address or to contain the actual content of a hypermedia Node. LocSpec is a class modeling the location specifier concept introduced in [13]. A LocSpec is used to specify a location within content specified by a ContentSpec; a LocSpec thus corresponds to the anchor value in the Dexter Model [18]. LocSpecs contain a number of optional attributes which can be filled with information to locate the same "object" in many different ways, e.g. by direct object reference, by the actual selection, by the surrounding context, and finally for special datatypes an n-dimensional axis specification inspired by HyTime [10]. The redundant location information can be used to heuristically detect and repair anchors in documents that have been changed on a server without notifying the external structure server, see Section 6.

2.1.1 Examples of LocSpecs for various media

Different media use different location characteristics. In what follows, we propose locSpec

attributes for some common media types; similar proposals were made in [19].

Text: The locSpec attributes for a span of text that can be marked as a persistent selection are as described in the previous section:

- Reference: a bookmark ID
- Selection: the text of the span to search for
- Axis specification: character count axis with a start position, and a length of the selection

Object drawings: Individual objects in, say CAD drawings, usually possess a unique identifier that the application uses to locate the object within internally maintained coordinate spaces. Since object IDs are usually unique and unaffected by edits related to other parts of a drawing, no redundant information is necessary to ensure link consistency. The locSpec simply consists of:

- Reference: ID of a built-in graphical object

Bitmaps: Anchors corresponding to rectangular areas of bitmaps can be represented in a locSpec using coordinates, for instance, the upper left and lower right corners of a rectangle. Some bitmap editors, however, support placement of objects in a layer on top of the bitmap. If these objects possess identifiers, they can be used as locSpec References as follows:

- Reference: ID of a graphical object in a transparent layer
- Axis specification: x and y axis with coordinates of selected regions

Video/sound: In time-based data, segments of video or sound are the typical units for anchoring. They are usually represented using a time offset from the start of the segment and a duration time or in terms of the number of frames. For video, another option is to overlay graphical objects on the running video at certain positions within the dimensions of the frame. Again, an identifier of the objects serves as the Reference attribute in the locSpec as follows:

- Reference: ID of a graphical object in a transparent layer
- Axis specification: a time axis with a segment specified by its start time and length or a number of frames axis with start and end frame numbers

Database records: An anchor in a relational database table can locate a span of several records, just as an anchor in text can span several lines or paragraphs. The actual query used to find particular records can also be used as the location method. In a locSpec, a query may simply be stored in the selection attribute. Hence, we have the following two location attributes for database records:

- Reference: key values for the records
- Selection: text of a query to find the records

2.1.2 Openness of the datamodel

The OHP data model is open in several respects. The specified classes may be specialized into new classes. Objects instantiated from these new classes may be associated with existing objects or new objects. Attributes can be added to existing types of object through the general Characteristics set in HMObject, which is a set of attribute-values. Computations and PSpecs can be associated to arbitrary hypermedia objects. At the same time the model provides a degree of typing that allow a shared and common semantics across hypermedia services.

In the following sections the architecture and the user interface to impose such structures on Web documents and other standard applications is discussed.

3 THE WEBVISE USER INTERFACE

This section gives a short introduction to the Webvise user interface and functionality.

The users can be provided with access to external structures in a variety of ways [3], see Figure 3. This includes a simple read only interface provided via a server [7] over light weight applets [15] to applications like Webvise with varying complexity and integration with authoring tools.
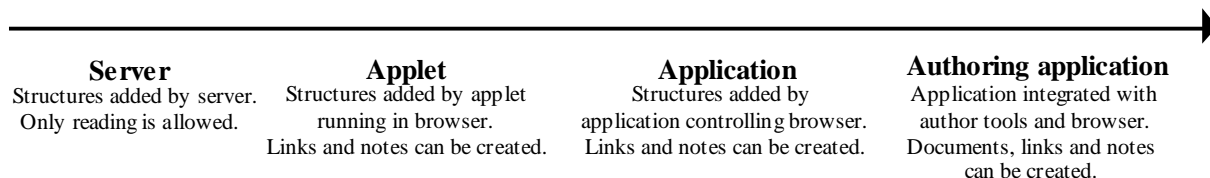
| **Server** Structures added by server. Only reading is allowed. | **Applet** Structures added by applet running in browser. Links and notes can be created. | **Application** Structures added by application controlling browser. Links and notes can be created. | **Authoring application** Application integrated with author tools and browser. Documents, links and notes can be created. |

Figure 3: Different degrees of control over external structures

In this section we will concentrate on the Webvise Client user interface and briefly describe the similar read only server interface. The Webvise Client user interface consists of two main components: an ordinary desktop application, which is the Webvise Client program and application extensions written for the integrated applications.
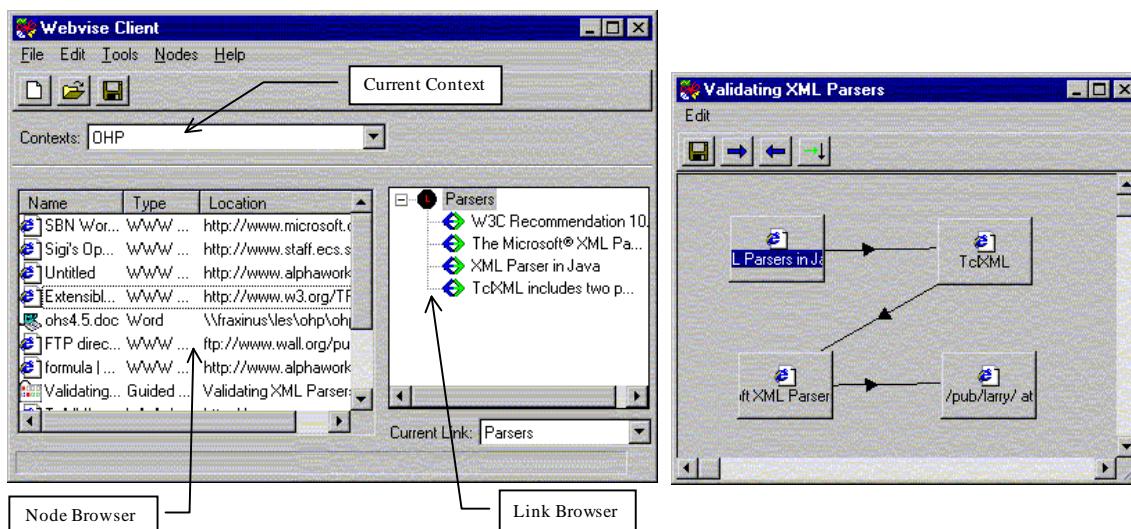

Figure 4. To the left the main window of the Webvise Client application. To the right the Guided Tour window with a guided tour about validating XML parsers.

3.1 The Webvise Client.

Figure 4 shows the user interface of the Webvise Client application. It allows the user to create and edit contexts that are available through the *Contexts* listbox. The nodes contained in the *current* Context are listed in the Node Browser. Double clicking on a node in the node browser activates the associated application and displays the document. To display the links pointing to or from a given node the user must right-click the node and select Links|Show Links from the pop-up menu. The links are displayed in the Link Browser. Nodes, links and anchors are referencing Web pages, Word documents and Excel spreadsheets are created from Internet Explorer, Word and Excel respectively, as described in section 3.2.

Guided Tours [30], [20] are created from the Nodes menu in the Webvise Client. A Guided Tour is a composite node, i.e. it contains other nodes and a graph through which these nodes are connected. The right window in Figure 4 shows an example of a small guided tour

through four Web sites with information about different validating XML parsers. A node is added to a guided tour by copying a reference to the node from the Node Browser window.

3.2 Open hypermedia extensions in standard applications

This section describes the extensions made to MS Internet Explorer, MS Word and Excel. The Internet Explorer is extended with a context menu (as shown in Figure 5.) to create external structures such as anchored links [18] and keyword based global links[2] and notes.

3.2.1 Anchored Links

To create an anchored link in the Internet Explorer, the user simply selects the text that should be anchored with the link. The user right-clicks with the mouse within the selection to pop up the context menu. From the context menu select "New Link". This creates both an anchor and a link with the new anchor in its first endpoint. The anchor will dynamically be marked up as a traditional embedded HTML anchor.

To create an anchor and associate it to some existing link, select some text in the Web page where you want the anchor to be located. Within the selection right-click with the mouse. From the context menu select "Add Anchor". External anchored links are followed the same way traditional embedded HTML links are followed.
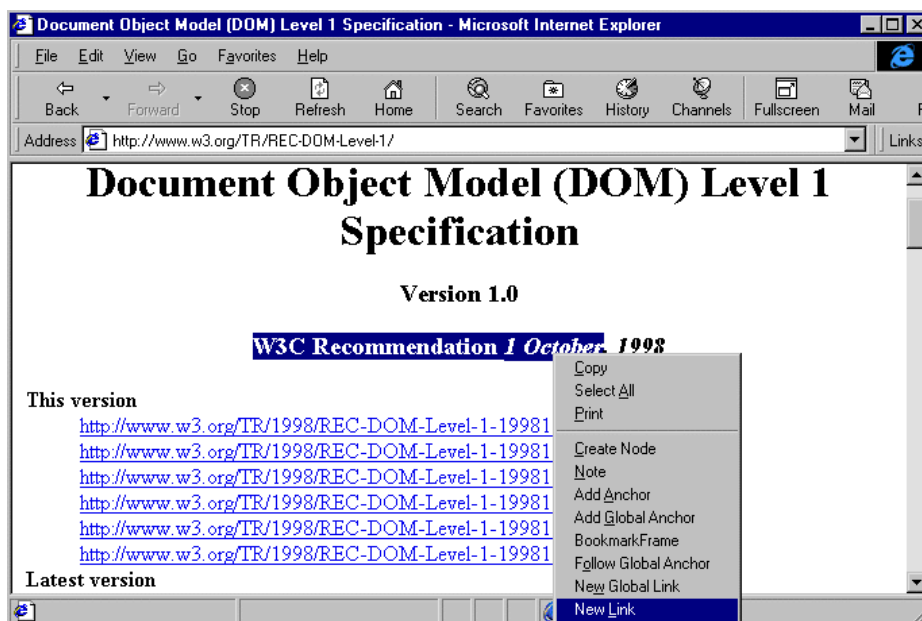

Figure 5. Microsoft Internet Explorer extended with open hypermedia services.

3.2.2 Global Links

We have introduced the notion of a global links, which is a generalized many-to-many endpoint variant [13] of Microcosm's generic link [19]. To create a global link, in the Internet Explorer the user selects the text of the *destination* anchor. Next, the user right-clicks within the selection. From the context menu "New Global Link" is selected to create both a global link and a destination anchor which is marked by changing the text color (pt. to green).

To create a *source* anchor (and associate it with an existing global link), select each of the

---

[2] A generalized version of the Microcosm generic links [19] with the posibility for more than two endpoints.

7

word(s) that should be associated with the source anchor. Within the selection of a given word, right-click with the mouse and choose "Add Global Anchor" from the context menu to associate it with the currently active global link.

To follow a global link, select the word(s) you want to investigate, then right-click with the mouse within the selection, and select "Follow Global Anchor" from the context menu. If the selected word(s) at an earlier point in time has been associated with a source anchor in a global link, the destination anchor of that global link will be displayed.

3.2.3   Notes

The Webvise extension also supports attachment of small notes to some piece of text in a Web page. These notes are similar to post-it notes known from traditional office working environments. To attach a note to one or more words in a Web page the user simply selects the text, brings up the context menu, and select the "Note" menu-item. This will bring up a small dialog where the note text can be typed. The note is implemented as a special type of anchor, and is marked up as a traditional embedded anchor. To read and/or modify the text in a note the user clicks on the anchor in the Internet browser to pop up the note dialog.

3.2.4   Linking in office applications

In Microsoft Word and Excel the open hypermedia services are available to the user via a toolbar and a menu. An example of the extension to MS Word is shown in Figure 6.

Microsoft Word has been extended with externally anchored links and global links. Arbitrary spans of text in Word documents can be used to anchor links externally. Like Word, Microsoft Excel has been extended with a Webvise toolbar. In Excel only anchored links are supported. The anchors can locate ranges of cells in a worksheet. By means of these integrations of standard office application a user can construct structures that combines local documents with Web-based documents.

Figure 6. Microsoft Word extended with open hypermedia services.

3.3   The Webvise Proxy server

Structures created by the Webvise Client can be accessed in *arbitrary* Web-browsers via a proxy server interface. This approach is taking the ideas of the Microcosm DLS [7] a step further and supporting the richer set of structures supported by the Webvise service. The

Webvise Proxy is developed with the building block framework described in [33]. If a user sets the preferences of the Web-browser to always go through the Webvise proxy, the proxy server supports the following features:

- Insertion of HTML <A> - tags representing the open hypermedia link endpoints created with the Webvise Client
- Insertion of JavaScript code to enable display of notes created with the Webvise client in a pop-up dialog on top of the text being annotated.
- Re-creation of the frame hierarchy in which the documents being linked and annotated were shown when the structures were created.
- Inserts a floating icon (currently a Coconut project logo) in the bottom right corner of the page, see Figure 7, which is linked to the configuration page for the proxy server.
- Look up of global link endpoint based on a text selection (Via a right-click menu item in Internet Explorer and via forms in other browsers)
- Integrated with the Ariadne guided tour applet [20] to present guided tours created with the Webvise client.

The proxy checks the Webvise server for every document being browsed in the Web-browser to find potential external structures to be imposed on the document. If such structures are found, then they are compiled into the document as listed above. The user is constantly reminded that s/he is running via the Webvise proxy, by the small floating icon which is inserted in the bottom right corner of each page, see Figure 7.
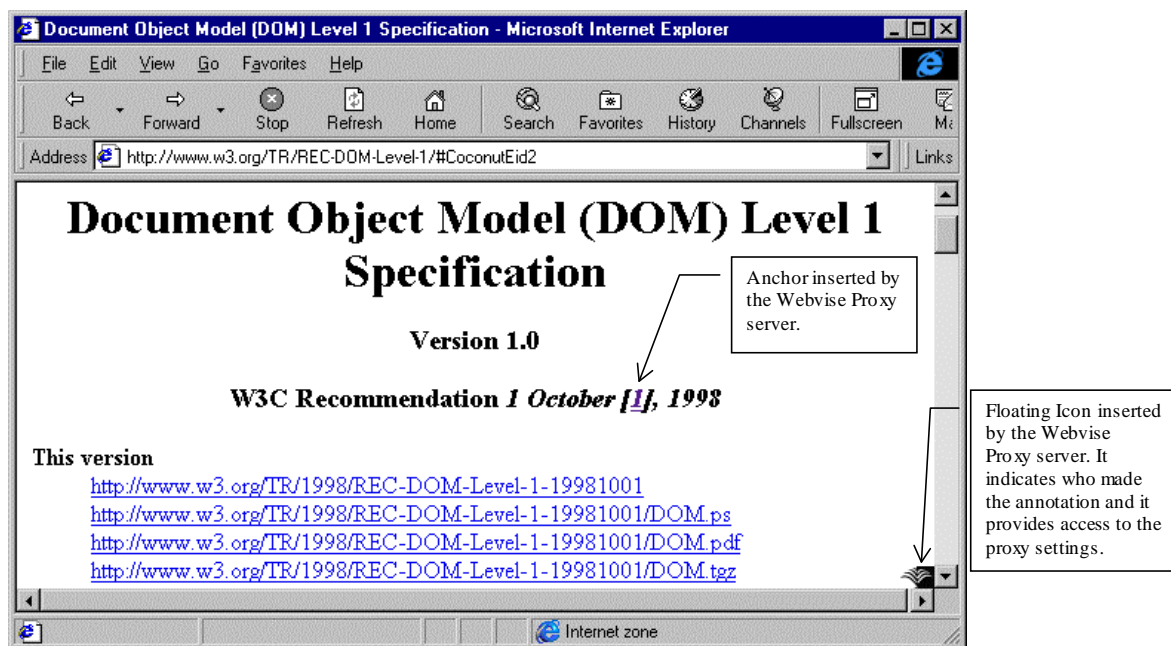

Figure 7: Webvise Proxy based interface to external structures. The link created in Figure 5 is shown by the proxy server.

Currently the proxy does not support links to and from local non-Web documents like Word and Excel documents, which can be interlinked using the Webvise Client. These documents needs to be published via a Web server, downloaded and handled via a Webvise Client.

4 ARCHITECTURE OF THE WEBVISE OPEN HYPERMEDIA SERVICE

This section describes the overall architecture of the Webvise open hypermedia service as well as a more detailed discussion of the architecture of the client application.

The Webvise Client enables the end user to take advantage of open hypermedia services when working with arbitrary applications. It does so by being a kind of "middleware" between various applications running on the user's workstation and one or more structure servers [25], [26], running on a server machine in the network, see Figure 8. In Open Hypermedia terminology, the Webvise application is an example of an Open Hypermedia client [27]. At this stage the Webvise Client has been integrated with Microsoft Internet Explorer 4.x, Microsoft Word 97 and Microsoft Excel 97[3] via the COM interface. The Java based Navigational Applets (NavLets) are described elsewhere [3].

The communication between the Webvise Client and the structure server(s) is compliant to the *Open Hypermedia Protocol - Navigational Interface (OHP-Nav)* [27], as depicted in Figure 8. This protocol is currently implemented as a textual protocol over TCP/IP, where the messages are encoded using XML [4].
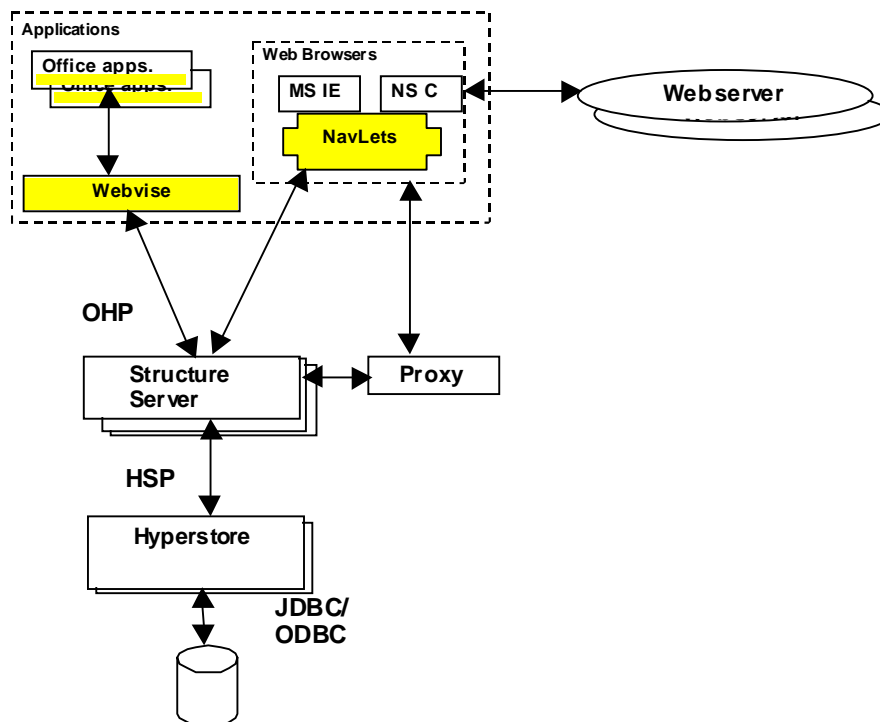

Figure 8: The architecture of the Webvise open hypermedia service.

4.1 Internal Structure and Implementation of the Webvise Client

This section will give a description of the internal structure of the Webvise Client, and the technology used to expose open hypermedia functionality to standard end user applications, explicitly on the Windows 95/98/NT platform.

The central component in the Webvise Client is the *Core*, which is responsible for:

1. Managing the set of ongoing open hypermedia sessions.

2. Managing the application wrappers.

3. Expose a user interface for open hypermedia services to standard end user applications.

Webvise handles communication with the structure server via client library API provided.

---

[3] Microsoft Word, Microsoft Excel and Microsoft Internet Explorer 4.x, ActiveX are either trademarks or registered trademarks of Microsoft Corporation.

The client library implements the operations needed to:

1. Send requests to one or more Structure Server(s).

2. Receive notifications from Structure Servers.

For each application that is extended with open hypermedia services, the Webvise Client implements a corresponding *application wrapper*. An application wrapper handles the communication with its corresponding application. This includes:

1. Launching the application.

2. Requesting the application to open and display (a particular part of) a document.

The creation and presentation of anchors within documents handled by standard end user applications is very much dependent on the openness of these applications. On the Windows platform many applications expose interfaces to parts of their features through Automation (formerly called OLE Automation). This allows other applications to access and manipulate the objects exposed. This is the technique used to integrate Webvise Client with MS Internet Explorer, MS Word and MS Excel, as depicted in Figure 9.

The Webvise Client exposes the open hypermedia services through the interface called *IWebviseApplObj*, which is implemented by the *Application Object*. The services exposed are a simplified subset of the open hypermedia services offered by the structure server. The services are simplified in the sense that standard end user applications can exploit some of the open hypermedia services without having to provide all the information required by the structure server, because the Webvise Client provides this information. The protocol used in the communication is an application specific subset of OHP-Nav [8], see Figure 8.
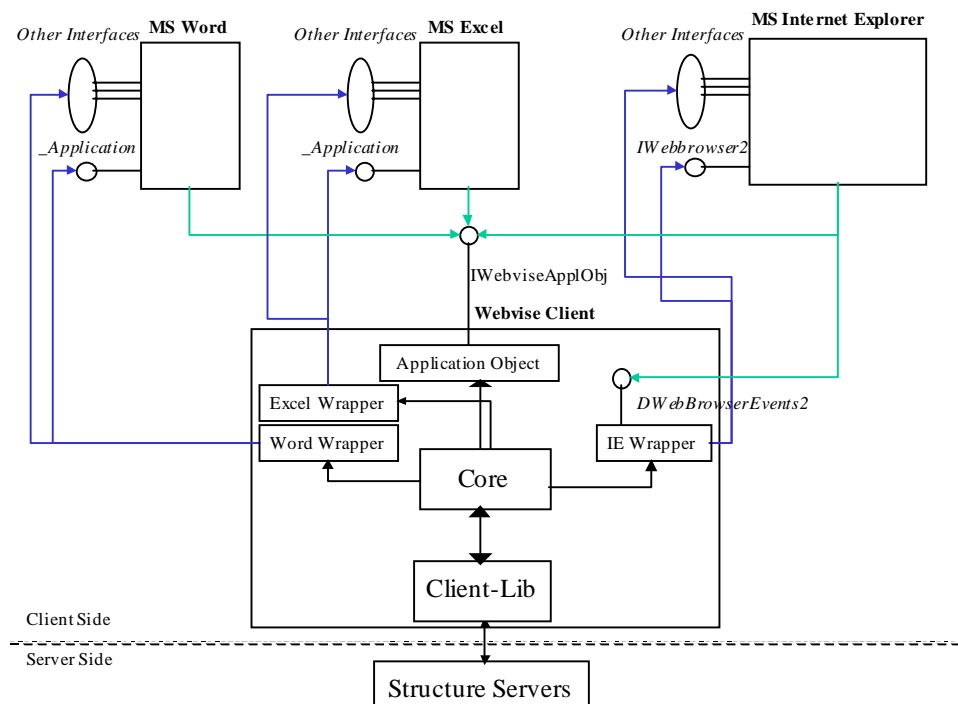

Figure 9. The internal structure of the Webvise Client, with focus on its integration/communication with standard end user applications.

The Microsoft IDL specification of the IWebviseApplObj interface contains operations such as NewLink, AddEndpoint, FollowLink, AttachNote, CreateNode, and OpenContext. This interface can be used by all applications that supports Automation. For more information on ActiveX, COM, Microsoft IDL and Automation [5], [29]. This application specific protocol

11

handles the requests e.g. for creating and following open hypermedia links made by the user via the Internet Explorer. In the future the small application specific OHP-Nav protocol might be implemented via other technologies, such as TCP/IP and CORBA [28], in order to support applications that prefer these technologies.

What makes the MS Internet Explorer well suited for this kind of integration are the interfaces to the Dynamic HTML's DOM [22]. In this model every HTML element is programmable and the interfaces give access to the objects and events in the object model. This allows the Webvise Client to request locSpec information from the HTML documents directly via the DOM. Moreover, the interface allows the Webvise Client to dynamically decorating the documents, in MS Internet Explorer with user created external structures such as links, anchors, and notes. It also allows Webvise to selectively show only some of the endpoints in a given document. The set of endpoints shown could be determined by the Contexts opened by a given user, or the access granted by the users who created the endpoint in a given document.

Finally, we note that the Webvise Client can be omitted in situations where applications themselves can be tailored to communicate directly to structure servers. But this is usually cumbersome, since each standard end user application then need to be augmented with code to implement several of the administrative tasks handled by Webvise Client. Also, it would require the application to communicate directly by means of OHP over TCP/IP, CORBA, RMI, DCOM etc. to structure servers. In the next section we will describe the appearance of the open hypermedia service in the user interface.

5 MANAGING INTEGRITY BETWEEN STRUCTURES AND CONTENTS

A central goal of open hypermedia is to maintain a consistent relationship between material managed by dedicated applications and hypermedia structures in a separate database. This problem is however hard to handle in general [19], [14]. In this section, we discuss problems of consistency and how they can be addressed heuristically.

5.1 Inconsistent anchors and dangling links

Inconsistencies between documents and external hypermedia structures may arise when documents can be moved, renamed, or deleted "behind the back" of the open hypermedia service. Furthermore, the documents may be opened and edited by versions of applications that are not integrated with the open hypermedia service. Consequently, the changes made to objects and locations inside the documents are not communicated to the hypermedia service, which in turn leads to inconsistencies. In the Web infrastructure editing of documents will typically take place by the original authors on their home site with no connection established to an open hypermedia service, thus there is a potential danger for inconsistencies like the one people are experiencing often when they consult old bookmarks.

Entire documents can also become unavailable, for example, if the content is a file identifier, and the file is *moved or deleted* independently of the open hypermedia system. In this case, the follow link operations should catch the HTTP exception and pass it along to the user together with the externally saved attributes that can inform the user about what went wrong.

Hall et al. [19] and Grønbæk & Trigg [14] propose heuristics to detect and repair link and anchor inconsistencies. For example, the hypermedia service can store modification date and time, in order to inform the user if a document changes after the anchor information was last updated. The other heuristics involve storing redundant information that helps the user repair the inconsistency. The first approach involves maintaining a component attribute that records modification dates. The redundancy heuristic is directly supported by the locSpec design.

5.1.1 Using modification dates to detect inconsistencies

A strategy for detecting inconsistencies is to store each node's last modification date in an attribute. Generally, this date is the same or later than the modification date of the document corresponding to the node's content. If not, we can conclude that the document was updated since the last communication with the open hypermedia service, indicating a possible inconsistency with the external structures in the hypermedia database. However, on the Web many documents are generated dynamically, and their modification date may change without the document contents changing considerably, thus a different approach is needed. We propose a heuristic approach based on redundant information stored in the locSpecs of the datamodel, as described in the next section.

5.1.2 Detecting and repairing inconsistencies using locSpecs

LocSpecs use several attributes to hold information about locations inside a node's content:
- Reference
- Selection
- Selection context
- Axis specification.

In most cases *one* of these attributes is sufficient to determine a location inside a node's content. But to detect and propose repairs of inconsistencies between a document and its hypermedia structure representation, we can take advantage of redundant location specification. For example, a span of text in a word processor or HTML-editor that supports bookmarks and targets names is uniquely identified by a locSpec whose Reference is set equal to the text span's bookmark ID or HTML target name. However, if the user, without communicating to the hypermedia service, deletes the bookmark or target from the document, then the locSpec can no longer be used to locate the span of text.

Repairing the inconsistency is possible, however, if the hypermedia service requests sufficient information to fill in all the attributes of the locSpec [19]. The span of text in the example can be redundantly identified with a locSpec of the following form:
- Reference: a bookmark ID or a HTML target name
- Selection: the text of the span to search for
- Selection context: some surrounding text
- Axis: a position, such as start position, and length of span on a character axis

With such redundant information, the hypermedia service can detect and repair a variety of inconsistent situations. The following is two examples:

A: A user accidentally deletes a bookmark or an HTML target referred to by a locSpec.

In this situation, the open hypermedia service can trigger the following repair actions:
1. If the anchor locSpec's referred bookmark or target is missing in the document, notify the user that the bookmark or target with the given ID no longer exists.
2. Direct the word processor/browser to highlight the span of text corresponding to the locSpec's axis character counts, and tell the user that the old bookmark or target represented this span.
3. If the span corresponding to the axis is identical with the stored text string in the locSpec's computation descriptor, ask the user to confirm updating of the anchor with new LocSpec information.
4. If the span corresponding to the axis is not consistent with the stored text string in the selection, notify the user and ask whether s/he wishes to perform a search for a nearby occurrence of the text string.

5. If the search is successful, ask the user whether s/he wishes to update the anchor to locate the span of text in this new position.

B: A user edits a document while disconnected from the open hypermedia service.

In this situation, the open hypermedia service can perform steps 2-5 of the preceding procedure for *every* anchor locSpec registered for the document. In most cases such strategies enable the user to re-establish links anchored in documents that have become inconsistent with the hypermedia structures. However, if none of the redundant locSpec attributes captures the intended location of the link endpoint, the endpoint needs to be removed or rebuilt from scratch.

6 USE SCENARIOS AND APPLICATIONS

The open hypermedia approach immediately provides support for imposing several different link and composite structures on the same body of Web documents that a user may or may not own. At the same time it allows anyone to make links *into* and *from* arbitrary documents and frame hierarchies. The goals of introducing open hypermedia services like Webvise in a WWW context are to support better individual users as well as workgroups in organizing and commenting on materials published on the Web [15]. In this sections we discuss a few examples of applications of such services.

6.1 Site-Reviews

Some Internet-based newspapers provide Web site reviews or compares information from multiple sites. For this purpose the journalist can use a service like Webvise to create their own link paths between details of the Web sites being compared and reviewed. In this case the Journalist will typically use the Webvise Client interface and the readers of the electronic newspaper will be using the proxy server interface. Here the little floating proxy icon (cf. Section 3.3) will be the logo of the newspaper drawing the readers attention to the fact that the visited sites have been augmented by the newspaper's open hypermedia service.

6.2 Advisory services via the Internet

We are working with a local agricultural advisory service who are using the open hypermedia service to help farmers in understanding environmental directions and laws published on governmental servers. To provide this understanding, the information on the governmental servers is linked to explaining texts and alternative sites by means of the Webvise open hypermedia services. The advice may also be concerned with chemical production companies' directions on how to apply their products in the field. Also in this case it is typically the advisors who are using the Webvise Client and the farmers who are using the proxy server interface. However, the advisors may work collaboratively on the production of such advisory information.

6.3 Digital library usage in Web based school projects

A local library is providing a lot of online historical material to be used as sources for school classes in writing projects. With the Webvise open hypermedia service, pupils can be collaboratively writing their project reports in integrated word processors and Web browsers. This way the underlying structure servers and hyperstore are used to coordinate the pupils' writing such that they are not working on the same section at the same time etc. Moreover the project report can utilize the Webvise ability directly from a word processor into a part of a document sitting deeply buried in some frame-hierarchy on the library pages or some other relevant server. In this case the project reports may be reviewed using the Webvise

application and integrated word processors.

6.4 Web-digests

Electronic magazines and many organizations often provide digests or trails [6] of specific subjects of interest. With the Webvise hypermedia service this can be done both by means of the guided tours and by means of threads of links made through many different documents sitting different servers. The guided tours are available through a graphical interface in Webvise, but they can also be provided by in a light weight Applet interface, called Ariadne [20] combined with the Proxy interface supplying the anchored and global links.

7 CONCLUSION

This paper has described the Webvise open hypermedia service for the WWW, which provides a link, annotation and guided tour authoring interface seamlessly integrated with the Microsoft Internet Explorer. The external open hypermedia structures can, however, be accessed in arbitrary Web-browsers via the Webvise proxy server interface. The papers also discussed issues in keeping the external hypermedia structures consistent with the contents of documents being structured. The Webvise services are further developed in the COCONUT project (http://www.cit.dk/coconut) which is a joint project between Tele Danmark Internet and University of Aarhus, Denmark. Work is in progress supporting open hypermedia linking of multimedia contents through the JavaMedia framework and Microsoft's MediaPlayer. Moreover, work on standardizing the open hypermedia protocols and datamodels is taking place in the Open Hypermedia Working Group [27].

8 ACKNOWLEDGEMENTS

9 REFERENCES

[1] K. M. Anderson, R. N., Taylor, & E. James Whitehead. Chimera: Hypertext for Heterogeneous Software Environments. In *Proceedings of the ECHT'94 European Conference on Hypermedia Technologies* 1994 pp. 94-107.

[2] T. Berners-Lee et al., World-Wide Web: The Information Universe. Electronic Networking: Research, Applications and Policy 1(2), 1992.

[3] N.O. Bouvin, Unifying Strategies for Web Augmenting. In the proceedings of Hypertext '99 (The tenth ACM conference on Hypertext and Hypermedia). Darmstadt, Germany, February 21-25 1999, ACM, New York, 1999 pp. 91-100.

[4] T. Bray, J. Paoli, & C.M. Sperberg-McQueen, Extensible Markup Language (XML) 1.0, W3C Recommendation 10-February-1998 , http://www.w3.org/TR/REC-xml.

[5] K. Brockschmidt, Inside OLE 2nd Edition, Microsoft Press, 1995.

[6] V. Bush, As We May Think. The Atlantic Monthly, August, 1945.

[7] L. Carr et al. The Distributed Link Service: A Tool for Publishers, Authors and Readers. in Fourth International World Wide Web Conference : ``The Web Revolution''. Boston, Massachusetts, USA, 1995.

[8] H. Davis, A. Lewis, & A. Rizk, OHP: A Draft Proposal for a Standard Open Hypermedia Protocol. In *2nd Workshop on Open Hypermedia Systems*, Washington DC: University of California, Irvine, 1996 pp. 27-53.

[9] H.C. Davis, S. Knight, & W. Hall, Light Hypermedia Link Services: A Study of Third Party Integration. in European Congerence on Hypermedia Technology (ECHT '94). Edinburgh, UK.: ACM, 1994.

[10] S. J. DeRose & D. G. Durand, *Making hypermedia work: A user's guide to HyTime*. Boston/Dordrecht/London: Kluwer, 1994.

[11] D. Engelbart, Authorship provisions in Augment. in Proc. IEEE Compcon Conference. San Francisco, California: IEEE 1984.

[12] K. Grønbæk & R.H. Trigg, Design issues for a Dexter-based hypermedia system. Communications of the ACM, 37(2, 1994 pp. 40-49.

[13] K. Grønbæk & R.H. Trigg. Toward a Dexter-based model for open hypermedia: Unifying embedded references and link objects. in HYPERTEXT '96 – Seventh ACM Conference on Hypertext. Washington DC, USA, March 16-20, 1996.

[14] K. Grønbæk, & R.H. Trigg, *From Web to Workplace: Designing Open Hypermedia Systems*. MIT Press, Boston Massechussets (In press).

[15] K. Grønbæk, N.O. Bouvin, & L. Sloth, *Designing Dexter-based hypermedia services for the World Wide Web.* In proceedings of Hypertext 97–The Eighth ACM International Conference on Hypertext. Southampton, UK, April 6-11, 1997.

[16] K. Grønbæk, J. Hem, O.L. Madsen, & L. Sloth, Cooperative Hypermedia Systems: A Dexter-Based Architecture. Communications of the ACM, 37(2) 1994 pp. 64-75.

[17] B.J. Haan et al., IRIS Hypermedia Services. Communications of the ACM, 35(1). 1992 pp. 36-51.

[18] F. Halasz & M. Schwartz, The Dexter Hypertext Reference Model. Communications of the ACM, 37(2) 1994 pp. 30-39.

[19] W. Hall, H. Davis, & G. Hutchings, Rethinking Hypermedia: The Microcosm Approach. Boston: Kluwer Academic Publishers 1996.

[20] J. Jühne, A.T. Jensen, & K. Grønbæk, *Ariadne: a Java-based guided tour system for the WorldWide Web*. In proceedings of The Seventh International World Wide Web Conference (WWW7) Brisbane, Queensland, Australia, 14 - 18 April 1998.

[21] H. Maurer, Hyperwave – The Next Generation Web Solution. Addison Wesley: Harlow, UK. 1996.

[22] Microsoft. Document Object Model**,** Microsoft Corporation**,** http://www.microsoft.com/workshop/author/om/doc_object.asp,1998.

[23] J. Nanard & M. Nanard, Using Structured Types to Incorporate Knowledge in Hypertext, in Proceedings of ACM Hypertext'91. 1991. p. 329-343.

[24] T.H. Nelson, Computer Lib/Dream Machines. Mindful Press 1974.

[25] P. J. Nürnberg, J. Leggett, E. R. Schneider,. As we should have thought. *Proceedings of the Hypertext '97 Conference.* (Apr 6-11). Southampton, UK 1997.

[26] P. J.Nürnberg, J. Leggett, E. R. Schneider, & J. L. Schnase,. Hypermedia operating systems: a new paradigm for computing. In *Seventh ACM Conference on Hypertext - Hypertext '96*. Washington, DC: ACM 1996.

[27] OHSWG. Open Hypermedia Systems Working Group WWW site (Sep 98). Website: http://www.ohswg.org/.

[28] OMG. CORBA 2.2/IIOP Specification., http://www.omg.org/corba/c2indx.htm. 1998.

[29] D. Rogerson, Inside COM – Microsoft's Component Object Model. Microsoft Press, Redmond, Washington 1997.

[30] R.H. Trigg, Guided Tours and Tabletops: Tools for Communicating in a Hypertext Environment. *ACM Transactions on Office Information Systems 6*(4) 1988 pp. 398-414.

[31] R.H. Trigg, A Network-Based Approach to Text Handling for the Online Scientific Community. Ph.D. Thesis: TR-1346, University of Maryland, 1983.

[32] U. K. Wiil & J. Leggett. HyperDisco: Collaborative authoring and internet distribution. In *Hypertext '97,*. Southampton, England: ACM Press 1997 pp. 13-23.

[33] P. Ørbæk, P. Building Blocks for Dynamic Web Applications. Department of Computer Science, Aarhus University. Draft submitted for publication, 1998.

VITAE

Kaj Grønbæk is associate professor at the Department of Computer Science, University of Aarhus, Denmark. He finished his master's degree in 1988 and his Ph.D. in 1991 both from the Dept. of Computer Science, University of Aarhus, Denmark. His research interests are: Hypermedia; Multimedia; Computer Supported Cooperative Work (CSCW); Cooperative Design (system development with active user involvement, cooperative prototyping); User interface design; object oriented tools and techniques for system development.

Lennert Sloth is research programmer at the Department of Computer Science, University of Aarhus, Denmark. He finished his master's degree in 1992 from the Dept. of Computer Science, University of Aarhus, Denmark. His areas of interests are: Design and development of hypermedia technologies. Development of graphical user interface libraries.

Peter Ørbæk is assistant professor at the Department of Computer Science, University of Aarhus, Denmark. He finished his master's degree in 1994 and his Ph.D. in 1997 both from the Dept. of Computer Science, University of Aarhus. His areas of interests are: programming languages, semantics, network and infrastructure. He works as an infrastructure specialist for CIT - The Danish national centre for IT research.