

Dynamic Use of Digital Library Material – Supporting Users with Typed Links in Open Hypermedia

Klaus Marius Hansen, Christian Yndigeegn, Kaj Grønbæk

Department of Computer Science, University of Aarhus, Åbogade 34, DK-8200 Aarhus N,
Denmark

E-mail: {marius, yndigeegn, kgronbak}@daimi.au.dk

Abstract. This paper introduces a novel approach to supporting digital library users in organising and annotating material. We have extended the concept of open hypermedia by introducing typed links, which support: addition of (user-defined) semantics to hypertexts, user navigation, and machine supported analysis and synthesis of hypermedia structures. The Webwise open hypermedia system is integrated with the World Wide Web, and has been augmented with a type system. We illustrate the potential use in the context of digital libraries with a scenario of teachers jointly preparing a course based on digital library material.

1 Introduction

Digital libraries may be viewed as a digital extension and enhancement of traditional libraries. The extensions and enhancements take place along several dimensions including the collections of the library, access to material, management of the library collection, and communication about the items in the collection [36]. Thus, a digital library can be seen as a collection of both transient and persistent documents that allows for work ranging from individual to collaborative work. Digital libraries may also be seen as a kind of hypermedia in the line of Bush's visions on creating a machine (the Memex) to help humans handle scientific literature [6]. Hypermedia technology is by many researchers viewed as a potential, powerful technology for use in implementing digital library systems [12], [22], [30], [38]. The World Wide Web (WWW) [3] hypermedia system is thus playing an important role in the implementation of most digital library systems emerging today.

This paper focuses on the application of hypermedia in the context of digital libraries. However, we aim at adding services to existing digital libraries rather than at constructing a hypermedia infrastructure for digital libraries per se. In doing this, we focus on approaches to applying open hypermedia to the problem of organising and annotating digital library material. Supporting digital library users in organising and annotating material found through querying of digital libraries is a problem domain that has been addressed by many digital library researchers [20], [25], [34]. This

paper addresses the problem faced by digital library users when having to use digital information for specific tasks.

1.1 The WWW and Digital Libraries

The WWW is emerging as an infrastructure implementing part of the universally available body of information envisioned by the early hypertext pioneers. Nevertheless, the WWW still suffers from a number of problems. These include that:

- Users need to own pages in order to link from them,
- making new anchors in documents requires these documents to be changed,
- collaboration on live documents is not supported, and
- no other structures than documents and links exist.

Adding to this the sheer size of the WWW, the classical problem of "being lost in Hyperspace" [8] is magnified. This means that the WWW per se is insufficient to support dynamic annotation and organisation of digital library material.

1.2 Open Hypermedia for the WWW

The work presented in this paper build upon approaches to integrate the WWW and open hypermedia [15], [16]. The Webvise open hypermedia service [15] provides structures such as contexts, links, annotations, and guided tours, stored in hypermedia databases external to the Web pages. In this way, Webvise supports users in creating links from parts of Web pages they do not own, and to parts of Web pages without writing HyperText Markup Language (HTML) [21] 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 break due to modified Web pages. Support for providing links to/from parts of non-HTML data, such as sound and movie is possible via interfaces to plug-ins and Java based media players [4].

The hypermedia structures are stored in a hypermedia database, based on the Devise Hypermedia framework [4], and the service is available on the WWW via an ordinary Uniform Resource Locator (URL). The best user interface for creating and manipulating the structures is currently provided for the Microsoft Internet Explorer browser pages (see Fig. 1) through Object Linking and Embedding (OLE) [5] integration that utilise the Explorers Document Object Model (<http://www.w3.org/DOM/>) representation of Web. 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 (called contexts) 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. Webvise clients are communicating with servers using the most recent version of the so-called open hypermedia protocol (OHP) [9], which is being developed by the open hypermedia systems working group OHSWG (<http://www.ohswg.org>).

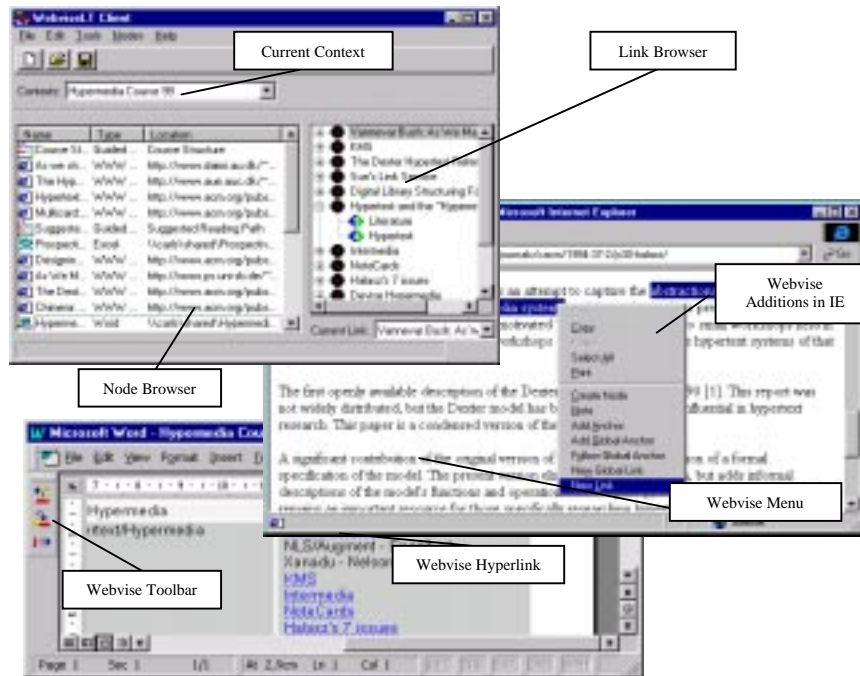


Fig. 1. Screenshot of Webvise Client User Interface.

1.3 Link Types and Digital Library Material

Trigg [37] introduced the notion of link types in hypermedia. Link types were to add semantics to the relationships between nodes in a hypertext. In Trigg [37], link types were semantic the relationships that researchers were creating when reviewing and discussing electronic scientific literature. In successor systems, such as NoteCards [17] and MacWeb [27], link types has been used for structuring of many other kinds of information.

An important aspect of introducing link types is that it supports the author of hypermedia links in expressing his/her intension with a specific relationship, but equally important it can support the reader in filtering and sorting the link relationships according to the link types assigned. Following the original ideas of link types in hypermedia they have many potentials in the area of digital libraries which should support users not only in finding material of interest but also to continue using the found material for individual or group purposes. Here a hypermedia system supporting link types will improve the digital library user in dynamically organising, commenting and discussing material with colleagues. This paper demonstrates an approach to construct an open hypermedia system supporting link types generally available to users who access digital libraries through the WWW.

1.4 Structure of the Paper

The structure of this paper is as follows: In section 2, we discuss link types and their usage in hypermedia and digital libraries. Section 3 and 4 discusses requirements for the implementation of typed links, and presents a prototype, called WebwiseLT in this paper, based on the Webwise open hypermedia system. Then, Section 5 discusses the use of WebwiseLT in a digital library use scenario. Section 6 discusses future work, and section 7 concludes the paper.

2 The Notion of Link Types in Hypermedia

This section introduces the notion of link types in hypermedia in general and relates it to other relevant areas of research.

Link types, or more generally types for, or knowledge of, objects in hypermedia structures, have been proposed as a means for reifying manageable conceptual models in hypermedia structures [27]. Having typed links may thus be useful in a number of ways, including that link types:

- add (user-defined) semantics to hypermedia structures,
- reduce user disorientation, and
- facilitate machine supported analysis and synthesis of hypermedia structures.

Links in a hypermedia structure provides structure between nodes allowing users to navigate. A binary link can be unidirectional, only allowing the user to jump from a node A to a destination B, but some systems provides bi-directional links which can be traversed in both directions. In general a link may have several destinations and the link itself may be traversed from any one to any other of these. Thus we can speak of physical direction of a link, expressing the navigational (and in some sense temporal) relationship between nodes, telling the reader how to navigate through the hypermedia structure.

With untyped links, regardless of which of the previous forms it is in, it is left to the anchor context to explain the purpose of the link, i.e. why it was created, and what the reader may expect to find at the other end of the link. Considering links between two nodes, types generally allows for a semantic interpretation of the relationship between the nodes, e.g. 'A comments on B', thereby giving the possibility of expressing knowledge about the link as a property of itself rather than solely by the context in which it is placed.

The type of the link describing a (binary) relationship between two nodes often implies a semantic direction as well as a physical. As pointed out by Trigg [37] these need not be the same. If a node B comments on a node A, the author of a context might want the reader to follow a link to a node B only after having read node A. The intended physical direction of the link would then be from A to B, whereas the semantic direction would be from B to A, expressing that "B is a comment on A". The distinction between physical and semantic direction can cause problems interpreting the relationship between nodes when the two directions differ. In the example above we have a physical direction from A to B. However, the semantic direction of the 'Comments on' link type has a semantic direction from B to A. In systems allowing

only unidirectional links, in which the physical direction of links do not change, the problem can be solved by the author making sure that the physical and semantic directions are the same when creating the context. The directionality problem increases if the system allows bi-directional links because the physical direction depends on which node is selected as source-node, whereas the semantic direction is fixed. This may lead to a situation in which both nodes could be interpreted as e.g. a comment to the other. Ideally, in this case the system should offer a mechanism to reverse the semantic direction of the link, depending on the physical direction (especially if nodes are not typed in the same manner as links, thus providing the reader a valuable help in determining the relationship between them).

The Dexter Hypertext Reference Model [18] proposes general bi-directional and n-ary links. Supporting such links raises a number of new issues on directionality and types. N-ary, unidirectional links with only one source-node but several destinations behave much like binary links (if the relationship between the source and each of its destinations is the same). In this case, the author could provide a link from a node A to nodes B, C, and D commenting on A. However, if there is not the same relationship between all the nodes of the link, the type of the link is restricted to express the most general relationship. This relationship is a common supertype if the types form a hierarchy. N-ary, bidirectional links make a special case. The link then connects a set of nodes with no fixed set of source nodes, and so a type can only express the common relationship between all the nodes of the link. In this case the semantic direction does not matter, as the semantic interpretation will be the same no matter in which way the link is traversed.

2.1 Link Types in Classical Hypermedia Systems

A taxonomy of link types for use in the hypertext system TextNet was presented by Trigg [37]. TextNet was designed to aid in creating and reviewing scientific papers. The taxonomy was accordingly fixed. The taxonomy consists of more than 80 types, mainly divided into 'Normal' and 'Commentary' link types. Examples of the link types belonging to the 'Normal' category intended for providing the connection between nodes in the text includes 'Citation' and 'Argument' (both being divided into further subtypes), as well as 'Formalization/Application' (depending on the direction of the link). The types in the 'Commentary' category are intended to connect statements about a node to the node, including 'Comment', types concerning style. Trigg notes that a user-extensible type system is conceivable, but argues that the taxonomy is extensive enough to cover any use of the system within the boundaries intended. Additionally, Trigg raises three practical issues in allowing dynamic type systems: Explosion of link types, reader confusion and system confusion.

Explosion of link types covers the situation in which the type system grows unmanageably as a result of modifications made by one or more users of the hypertext system. This is partially connected to the second consideration, that of reader confusion. If users are allowed to extend the type system incrementally, it is very likely that they will not fully understand the modifications made by others. This may mean, that they will add new types or modifying existing ones to cover what other users has already expressed, or misusing types for something they were not intended

for. System confusion, the third consideration, has to do with the relationships between the hypertext system and the link types. Some systems may have a partial understanding of the semantics of the type systems. In this case, the users may be required to give information to the system concerning new types.

It will however almost certainly be necessary to allow user-configurable type systems in general purpose hypermedia if one wishes to allow typed links. If the domain in which the system will be used can not be determined in advance, then neither can an appropriate type system. It is then up to the users of the system to handle explosion of link types and reader confusion.

Several systems have a notion of typed links. JANUS [11] is an example of such a system in that it is a combination of an issue-based hypertext system supporting argumentation and a knowledge-based system. Issues in the underlying hypermedia system are connected to each other by relationships like 'is more general than'. NoteCards [17] is an example of a system allowing typed links with a semantic direction. A type is simply a label selected by the user to describe the relationship between the source and destination node. In this way, NoteCards allowed user-defined types. Consequently, it can in this respect be seen as an extension of the principles in TextNet, with a dynamic type system to allow for user-customisation with regard to specific usage.

The type systems mentioned above can be extended from being 'merely' typed for the purpose of semantic reading to be type systems which allow link types to have (user-defined) attributes as well as scripts attached to its types. In [27], an example of an object oriented model, implemented in MacWeb, is given. In such a model a link type can define attributes as well as scripts, and it can inherit attributes and scripts from multiple supertypes, giving the hypermedia structure the potential of being used in a wide range of use scenarios. The type system is implemented and modified in the hypermedia structure along with the contents itself.

2.2 Link Types on the WWW

In the case of the WWW, HTML allows the marking of a region in a node as being a link to an entire node or into an anchor in a node. In HTML links are unidirectional, and only supports one source and one destination anchor. They are also embedded in the documents they link. The most common use of a link in HTML is to fetch another document. However, the suggested HTML+ standard [32] made a proposal for the embedding of types into links. Types were to be embedded in links and the type of a link should be specified via the 'rel' and 'rev' attributes of a link. Some predefined types for processing of documents existed, while user defined link types were also made possible. The current HTML standard proposal [21] contains a subset of the HTML+ specification on link types.

2.3 Link Types in Open (Hypermedia) Systems

Many open hypermedia systems have been implemented, e.g., Chimera [1], Microcosm [10], DHM [13], HyperWave [26], and MultiCard [33], but none of them

support a link type system coming close to that of TextNet or NoteCard. Microcosm supports different link behaviours for what is called specific links, generic links, local links, and retrieval links. However, these different link behaviours do not correspond to any semantic link types as described above. Some of the systems, e.g. DHM, supports the addition of general user-defined attribute/value pairs to objects in hypermedia structures. This means that, in principle, a simple way to provide types is provided: Users or implementers may use the attribute/value pairs to encode a type of a hypermedia object. However, no use has generally been made of this facility with respect to providing types. Thus, to our knowledge, no open hypermedia system has implemented thorough support for link types, including hierarchical type systems and editors and browsers for link types. The 'ComMentor' architecture [34] provides means for annotating Web pages. This facility may be used in a similar way as attribute/value pairs to simulate types of links.

2.4 Types in Programming Languages

Since the 50's, types have been an integral part of many programming languages [35]. In programming languages, a type is a set of entities having some properties in common. An example of a type is 'integer': Integers may be added, multiplied, and so on, whereas integers cannot be compared to strings. Any entity belonging to the set of 'integers' is guaranteed at least to have the properties of integers. Thus, types in programming languages may be viewed as providing the user of an entity with a "contract" on the capabilities of that entity. Object-orientation [24] gives a highly structured view on types. Entities in an object-oriented program ('objects') are seen as instances of concepts ('classes'). These concepts are the types of (typed) object-oriented programming languages. Just as concepts in the real world may be structured by means of classification and composition, object-oriented programming languages support classification and composition of types. Types may be classified in taxonomies, or classification hierarchies, and types may be used in defining other types through composition. The notion of (link) types for open hypermedia introduced in this paper is inspired by the object-oriented view on types.

3 Designing and Implementing Typed Link Support for Digital Libraries

In this section, we first discuss the design of a link type system for open hypermedia. Then we discuss the integration of open hypermedia in digital libraries. Finally, based on this discussion, we describe the implementation WebviseLT, a prototype of an open hypermedia system extended with link types, based on Webvise.

3.1 Design of Support for Typed Links

As discussed, types can be seen as modelling concepts that cover relationships between components in the hypermedia domain that a concrete hypermedia structure is all about. Trigg [37] discusses scientific writing and mentions types such as comments and argument. Nanard et al. [27] discuss the use of types (in general) to represent knowledge and mention link types such as "Has Painted" or "Has Carved" for educational hypermedia structures. Ideally then, a design of typed links would incorporate means for at least classification and composition of data and actions. This disqualifies the simple types of e.g. NoteCard [17] (that are merely tags or labels on existing links). Also, if link types are to be used dynamically in daily use, several conditions must be met. First, it should be possible to have typed and non-typed links co-exist. Second, the type system should be so dynamic that it allows evolution while in use. The necessary dynamics may be achieved by, among others, allowing users to add attributes to links without changing their type, to change the type system without "breaking" the types of existing links, and to change the type of links dynamically. Then, if a general-purpose open hypermedia system, like Webwise, is to implement link types, it certainly should not implement a fixed type system, but rather a dynamic one specific to individual contexts.

We thus view types as evolving, external, and encompassing data and action: A link type has a set of data and actions, and link types can be specialised. As an example, in this conceptual view Trigg's 'Argument' link type is a specialisation of a 'Normal' link type. It might have an attribute 'Against'. The 'Normal' link type might have an action. Any link with type 'Argument' would then have an 'Against' attribute and the action of the 'Normal' link type.

3.2 Integrating Open Hypermedia in a Digital Library Architecture

Since our approach is to introduce typed links in digital libraries by means of an open hypermedia system, we need to describe how open hypermedia may fit in as a value adding service in arbitrary digital libraries accessible via the WWW. Open hypermedia systems as well as digital library systems may be roughly described as three-tier architectures as shown in Fig. 2 in a slightly modified version of Bass et al.'s [2] software architecture notation.

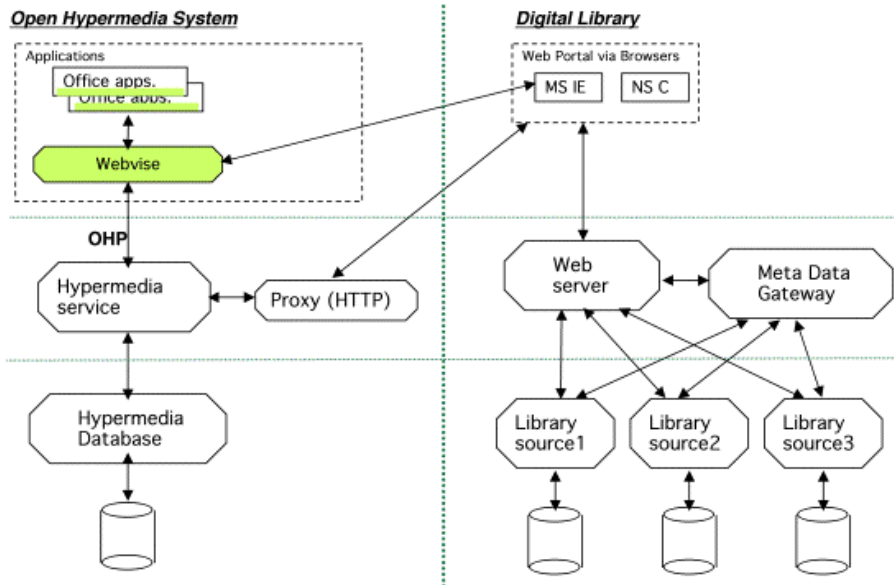


Fig. 2. Architectural relationship between open hypermedia systems and digital libraries.

The digital library material is accessed via a WWW portal in the architecture above. The portal may integrate a number of library sources through a common webserver and metadata gateway. The open hypermedia system can be integrated in an orthogonal manner and the hypermedia structures can be stored externally to the library sources. The open hypermedia service may be provided on the same server as the digital library portal server or on a separate server. The only assumptions are that the Web browser in question can communicate with the open hypermedia client (Webvise in this case) and that the addresses (URLs), provided by the WWW portal server, are stable.

3.3 WebviseLT Architecture and Implementation

Extending the storage layer of Webvise to implement link types would seem ideal: Webvise is written in the object-oriented language BETA [24], and the structure of our link types coincide with the structure of object-oriented types. This is, however, not a very good idea for several reasons. First, the structure of link types are most profitable created in situ, and BETA - or any other typed, class-based language - is currently not capable of supporting this in the generality that is demanded. Second, there is a general trend in (open) hypermedia towards standardisation (<http://www.ohswg.org>) meaning that directly extending the storage layer of Webvise, implementing link types and sub-types of the 'link' class in Webvise, would be problematic since it almost certainly would have to change. Third, types may best be viewed as external to hypermedia structures. In this way types may have an existence across different hypermedia (and hypermedia systems).

The implementation of WebviseLT is based on the current version of Webvise and runs on the Windows NT/9x platform. Fig. 3 shows how the architecture of WebviseLT fits into the general open hypermedia architecture model described above.

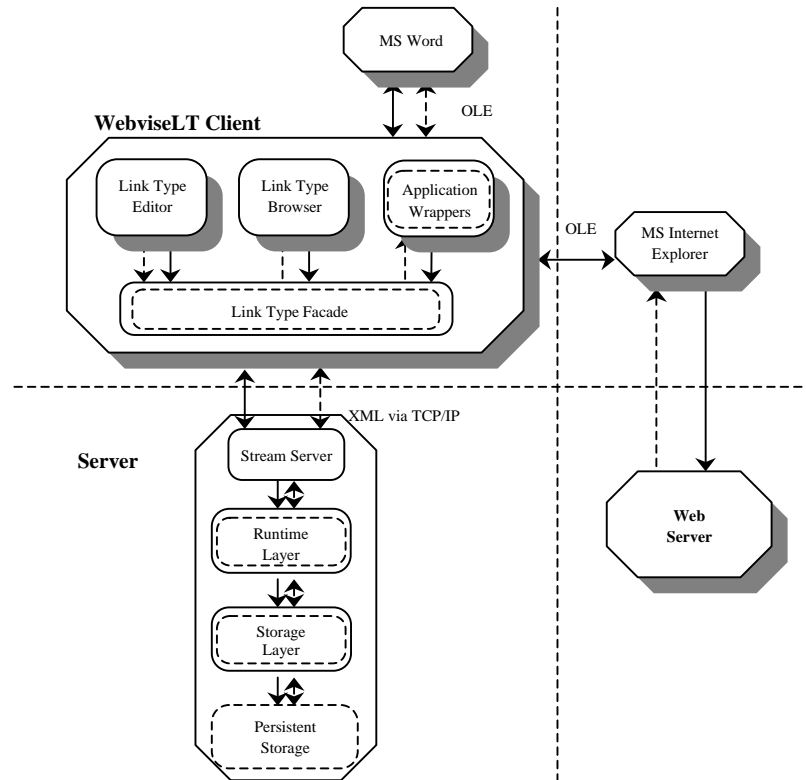


Fig. 3. Current Architecture of WebviseLT

The WebviseLT client is shown as a process communicating via TCP/IP with the Webvise server. The Webvise server implements a standard, layered open hypermedia system architecture. The WebviseLT client communicates with external applications such as Microsoft Word and Microsoft Internet Explorer using OLE communication.

Only subcomponents of the WebviseLT client that concern the implementation of link types are shown. Of these, the 'Link Type Facade' component is central for the extension of Webvise. The instantiation and associated use of the component is an application of the 'Domain Model Concealer' architectural pattern [19]. It handles all saving, loading, and interpretation of link types on the client side by effectively transforming its interface into calls to the server. This means that the OHSWG standard communication protocol used by Webvise has been slightly extended to cater for link types. Since the facade is the sole point of handling of link types in the client, changes to representation of link types are simple to make. This was seen as a

necessity since the server component of the Webwise system was actually changed to conform to the new standards of the OHSWG.

Scripting. A vital part of semantics is action. We make a preliminary attempt at providing this for link types by incorporating scripts written in TCL/TK [31] into links. Since attributes with the name 'Script' are interpreted as TCL/TK scripts, scripts are edited in the client in the same way that attributes are edited. An example of this is shown in Fig. 5, below. The script maintains a usage count ('noOfUsages') for the link and a date at which the link was last followed ('lastUsed') for any link with a type that is 'Link Type' or a specialisation thereof.

The current simple implementation executes the scripts on the client, and only when following links or presenting anchors, but more elaborate tailoring schemes are imaginable and desirable.

Representing Link Types. The directed acyclic graph of link types is saved in XML format [39] as an attribute on a context object. An example of this is shown in Fig. 4.

```
<linktypes>
  <linktype>
    <typeid>0</typeid>
    <typename>Link Type</typename>
    <typeattributes>
      <attrname>noOfUsages</attrname>
      <attrvalue>0</attrvalue>
    </typeattributes>
    <supertypes></supertypes>
  </linktype>
  <linktype>
    <typeid>1</typeid>
    <typename>Quality of Source</typename>
    <typeattributes>
      <attrname>Why</attrname>
      <attrvalue></attrvalue>
    </typeattributes>
    <supertypes>
      <typeid>0</typeid></supertypes>
  </linktype>
</linktypes>
```

Fig. 4. Example of XML Format of a Link Type System.

The example defines a link type system with two types 'Link Type' and 'Quality of Source'. 'Link Type' has a 'noOfUsages' attribute with the default value '0'. 'Quality of Source' also has a 'noOfUsages' attribute with the default value '0', since it is a subtype of 'Link Type'. Moreover, 'Quality of Source' has an attribute 'Why' with no default value. If a link has a type, the type is saved as an attribute on the link. This attribute has an id into the link type system as value. If the link does not have a type, such an attribute does not exist on the link.

By using XML format, the link type system can be represented and stored externally to contexts. This means that it becomes possible to exchange link type systems, not only between contexts, but also between different hypermedia systems.

More, combination of link type systems may also be facilitated. This may potentially lead to conflict with respect to e.g. naming. See section 6.2 for a discussion of this.

Integration with Other Applications. In order to use the typing mechanism efficiently in actual work, the functionality of WebviseLT needs to be integrated with commonly used applications such as Microsoft Word or Internet Explorer. This means, among others, that external applications need to be able to visualise types of links. As an example, the OLE communication from Webvise to Microsoft Internet Explorer was changed to pass on the name of the type for the link being presented. Currently, the name of the type is simply inserted as a 'title' on links [29]. This creates a tool-tip with the name of the link type when the mouse is placed over the link. In this way (part of) the knowledge incorporated in the link type is passed on to the user immediately.

If the link is not explicitly typed no type description will appear. Later we will touch upon what type of information to give the user when presenting links.

4 Dynamic Use of Link Types in WebviseLT

In this section, we discuss aspects of the creation and use of link types based on WebviseLT.

4.1 Tailoring the Type System

In WebviseLT, editing of link types is done in a graphical link type editor presenting the user with a view of the type system (Fig. 5.) Using the editor, a user may freely add and delete types as well as add and remove specialisations between types. For any type, a number of attributes and their default values may be specified. In Fig. 5 it is e.g. indicated that the basic link type ('Link Type') has five attributes ('Type name', 'Owner', 'LastUsed', 'noOfUsages', and 'Script'). This means that any link that has the type 'Link Type' (or a specialisation hereof) is guaranteed to have at least these five attributes. Furthermore, such links will have a 'Script' attribute with the default value shown in the window on top.

It is possible to change the type hierarchy after having assigned link types to links. This introduces the difficult problem of schema evolution and database reorganisation [23]: Since a type for a link may be viewed as guaranteeing a specific interface of a link, any editing of the link type hierarchy must preserve this property. In the current implementation any addition to the link type hierarchy that does not affect existing typed links is unproblematic. An example of such a change would be to add an 'Intermediate' type as a specialisation of the 'Required Knowledge' type. However, a change to the type of an existing typed link will affect that link. If the type is removed, the link will be untyped. If the type is changed, the link will have the type reapplied. Thus, the interface property is preserved.

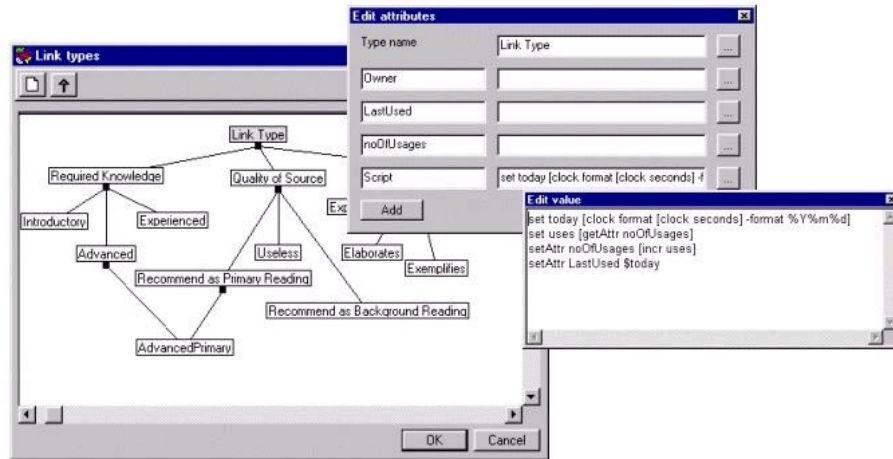


Fig. 5. Link Types Editor.

4.2 Dynamic Assignment of Types

Evidently the user should have a way to specify and inspect the type of a given link. This is accomplished through the tree-based type selection dialog. With a link of a given type the list will open with that type marked (Fig. 6.) If the link is not typed, no element of the list will be marked.

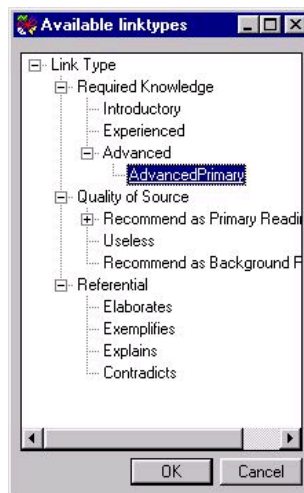


Fig. 6. Available Link Types Viewer and Selector.

A most general type will be the root of the tree and subtypes of a type will be shown on a branch node of a type's node. In the case of multiple inheritance for a given type it will appear under each super type. This may potentially lead to exponentially large trees, but it certainly gives the most accurate description of the relation between types. Upon accepting the dialogue, the selected link in the WebviseLT client is assigned either the chosen type or no type if no type was chosen. After the assignment of a type to a link, the user may freely add and remove attributes of the link not connected to the assigned link type (not shown). One might also allow the user to choose more than one type from the dialogue, thus allowing multiple inheritance for a single link to be declared on the fly.

4.3 Querying Contexts With Typed Links

A general query framework for contexts has been examined. This has been done using a (slightly changed) and simplified version of the Object-oriented Query Language (OQL) [7]. This general query mechanism has been used to implement a simple query mechanism for searching contexts in WebviseLT

The user interface is rather simple (Fig. 7). It allows for searches on hypermedia components such as links and anchors using a list of simple conjunctions or disjunctions. In Fig. 7, a context named 'Hypermedia Course 99' is searched for link components with type 'AdvancedPrimary' that has a usage count greater than 5.

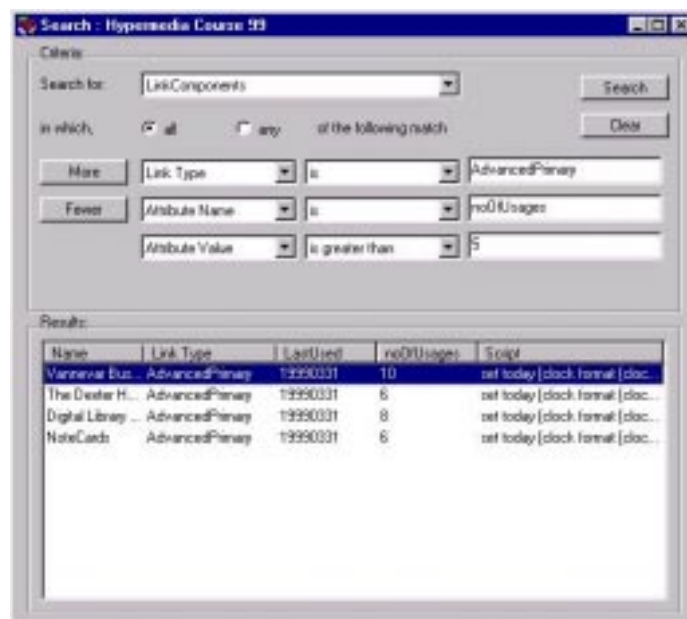


Fig. 7. The Search Dialog. Searching for Links with Type 'AdvancedPrimary' that have been followed at least five times.

By pressing "More", the user may include more search criteria. Pressing "Less" would mean that the search for "Attribute Value" that "is greater than" would be excluded from the search. This means that the search will be for links with type 'AdvancedPrimary' that has an attribute named "noOfUsages". The result of the current search is, as shown in the results canvas, four elements representing links. If such an element is double-clicked, the first node, to which the link belongs, will be presented in the node browser of the WebviseLT client and the link itself will be presented in the link browser of the WebviseLT client. In order to communicate the query, it is translated to an equivalent (simplified) OQL query and executed on the server.

5 Applying WebviseLT in Digital Libraries: A Use Scenario

This section describes a typical use of digital libraries in the planning of a new course based on digital library material. Open hypermedia can be used to organise digital library material for the specific purpose of a user. Organisation may consist in creation of collections, hierarchies, linked paths, and annotations. The notion of semantic types further enriches the means for organising material.

5.1 The Setting

Christiana is teaching at a school of library science and Miguel teaches at a computer science department. Together they are planning a new course in hypermedia that will have participants from both sites. Since their daily workplaces are in different locations, they decide to do the planning over the Web supplemented with phone and email conversations. They have access to both international and national libraries as well as publishers through a common Web portal. They also have access to the WebviseLT system, which enable them to share their planning documents and their collaborative structuring of the digital library material they find via the portal. Both Christiana and Miguel are creating documents on their desktop containing outline proposals of the course, lists of proposed readings, etc. The central documents and the type system they create are illustrated in the Figs 5-7, with a temporary plan as the central document with typed links to the different sources. This plan is placed on a server that both have access to.

5.2 Organising Material

When planning the specific sections of the course, the teachers issue a number of queries for relevant digital library materials. The results are organised with the WebviseLT system supporting link types. The dynamic planning documents are managed in MS Word extended with open hypermedia, and the library sources are accessed via an open hypermedia augmented internet Explorer. A temporary plan for the topics of the course is made in Microsoft Word after phone based discussions.

The two teachers start introducing link types to distinguish between the links they create to the digital library sources. They develop their own conventions about semantic link types similar to the types introduced by Trigg [37] for reviewing scientific literature. For instance, they introduce link types such as 'Introductory', 'Experienced', and 'Advanced' to distinguish sources based on the knowledge required to read them (Fig. 5). To facilitate the discussion of the quality of the sources, they introduce link types such as 'Recommend as Primary Reading', 'Recommend as Background Reading', and 'Useless'. As they are collecting source material, they create anchors for links in the planning document and links to the relevant sources. Link types are then assigned to the links (Fig 7) and used to communicate why the actual link to a document is included. As they continue to organise the material, links with multiple types are needed. To express, e.g., that a text that is 'Recommended as Primary Reading' requires a knowledge level of 'Advanced' in order to read it, Christiana creates a new link type 'AdvancedPrimary' that has 'Advanced' and 'Recommend as Primary Reading' as supertypes and assigns this type to the link. New link types are introduced as needed.

Both of the above sets of link types are meant to express assessments about the destination node as a whole. Also a number of types are used to provide meaning to the referential links between anchored segments of the material. These types are typically called 'Elaborates', 'Exemplifies', 'Contradicts', 'Explains', etc. Miguel may also decide to prepare the hypermedia structures for monitoring the students use of the material. To facilitate simple monitoring of the students' use of the context they create, Miguel adds attributes to all link types ('noOfUsages', 'lastUsed') and create scripts to process the use of the context (Fig 6).

5.3 Use of Material

The teachers use the type mechanisms to generate filtered browsers for their students, based on their own collection of resources. They decide to provide their students with filtered browsers for resources of the types 'Recommended as Primary reading' and 'Recommended as Background reading' for each week as well as the full length of the course. Their use of attributes and scripts enables them to get a picture of the usage of the online material. If e.g. they want to know how many link followings there has been to advanced material they have found "recommended as primary reading", they may perform a search as shown in Fig. 7.

As the contexts given to the students are identical to the teachers, except that useless resources are filtered out, the students have access to the referential links between papers as well as the notes made by the teachers. To some of their students, this is a nice feature. Although they may prefer to read the articles on paper rather than the screen, they will go through next week's context in WebwiseLT as part of their preparation. Doing this, they may find that the referential links and the occasional notes help in reading the papers. And as the course progresses, they may lift parts of the resources in to their own contexts in order to make some of their own thoughts of the papers and their relationship persistent.

6 Status and Future Work

This section first describes a brief status of the WebviseLT prototype and then we briefly discuss a number of possible extension to WebviseLT and the notion of types in open hypermedia in general. The approach to adding typed links to open hypermedia has been implemented in WebviseLT as an extension to Webvise. The current version of WebviseLT may be downloaded from <http://www.daimi.au.dk/~marius/hypermedia/webvise/>.

6.1 Typed Components

In this paper, we have primarily focussed on typed links. The above discussion about type systems for links generally applies to type systems for any kind of component in a hypermedia system [28]. In the object-oriented model of MacWeb, e.g., types for nodes can be handled in much the same way as types for links. While the type of links helps to identify the relationship between the source and the target(s) of the link, the type of nodes may help identify the classification of a node. This is probably even more so if the type system for links is 'weak', meaning a link of a given type may lead to more than one type of node. Such an extension would be fairly easy in WebviseLT, since the mechanisms used to handle attributes (of which the type it self is one) for links would work for all types of components.

In the same manner, WebviseLT could easily be extended to allow typing of its built-in pop-up notes. WebviseLT supports insertion and subsequent sharing of notes on arbitrary points of a webpage. When selected the note will be displayed in a dialog box. Typing of notes would, as is the case with links, provide the reader with additional information about the contents of the note, prior to following it, making it possible for the reader to avoid opening notes that are not of interest.

6.2 Combination of Contexts

We tend to believe that dynamic type systems will generally evolve into relatively closed type systems, which only rarely needs to be modified. If this is true, then we should certainly cater for the users, who creates new contexts every once in a while. We took the design decision of placing the type system in the context, rather than globally in the system, but it ought to be able to transfer or share type systems between contexts. This poses some problems, however. What should be done if a type system is imported into a context with an existing type system? Should it simply replace the old type system, or should the system try to merge them? This again introduces the problem of schema evolution that could be handled more generally than in the current implementation. The user might for example use domain knowledge to specify a graceful merge of type systems.

Another issue is importing one context into another. If the components of the imported context have been typed, how should this be handled? In particular, clashes of type names require human resolving.

6.3 Presenting Navigational Information

As previously discussed, it is necessary to visualise link types in external applications. When Webvise is further extended to allow other kinds of typed hypermedia components, this knowledge has to be displayed as well. Yet a possible extension would be to show the document type of the target, i.e. the application which will present the target node. This can only be done if there is only one possible target for a link, but it could be used in a list of possible targets as well. The benefit of presenting one or more of the above navigational information would be added predictability when navigating the context. Possibly the user could avoid following links which led to targets of no particular interest in the current use situation. However it would also mean an added overhead for presenting nodes, if all links in the node had to be resolved in advance to determine the attributes of the target(s). This is not a problem when confining the system to show only the type of the link, since this information is accessible anyway.

6.4 Typing Types

Although our implementation supports an object-oriented type description, the actual types are not stored in such a way at all. To gain the full strength of this approach we still have to consider a number of enhancements. Especially important is typing of attributes of types: Currently all attributes are considered as being of (data) type 'string'. An extension could also support composition and association between types, supporting the structuring mechanisms from object-oriented programming languages in full generality. Also, when the standardised server is in place real, strongly typed links via dynamic linking or interpretation could profitably be considered. This will also include extended and general scripting support.

7 Conclusions

This paper has introduced an approach to support digital library users in organising and annotating material found in digital libraries. This approach is based on open hypermedia. It has taken the open hypermedia approach a step further by introducing support for typed links in open hypermedia and in turn demonstrated the potentials for digital library users.

Link types in open hypermedia support: addition of (user-defined) semantics to hypermedia structures, reduction of user disorientation, and machine supported analysis and synthesis of hypermedia structures. A type specifies a set of attributes and methods are guaranteed to exist for links of that type, thus giving semantics to links. Moreover supporting users in defining their own types with corresponding semantics to links is important in the context of emerging structures. The Webvise open hypermedia system, which is tightly integrated with the MS Internet Explorer, has been augmented with a type system according to this approach. Finally, the

potential use of WebviseLT in the context of digital libraries has been illustrated with a scenario of teachers jointly preparing a course based on digital library material.

8 Acknowledgements

This work has been supported by the EU ESPRIT LTR Project 31870 'DESARTE', and the Danish Research Council's Center for Multimedia (project no 9600869).

9 References

1. Anderson, K. M., Taylor, R. N., Whitehead, E. James. (1994) Chimera: Hypertext for Heterogeneous Software Environments. In Proceedings of the ECHT'94, pp. 94-107.
2. Bass, L., Clements, P., Kazman, R. (1998) Software Architecture in Practice. Addison-Wesley.
3. Berners-Lee, T., Cailliau, R., Groff J.-F., Pollerman, B. (1992) World-Wide Web: The Information Universe. Electronic Networking: Research, Application, and Technology 1(2).
4. Bouvin, N.O., Schade, R. (1999). Integrating Temporal Media and Open Hypermedia on the World Wide Web. To Appear in Proceedings of the Eighth World Wide Web Conference 1999, Toronto, Canada.
5. Brockschmidt, K. (1995) Inside OLE. Microsoft Press International.
6. Bush, V. (1945) As We May Think. Atlantic Monthly, July.
7. Cattell, R.G.G. (Ed.) (1996) The Object Database Standard: ODMG-93 Release 1.2. Morgan-Kaufmann, San Francisco.
8. Conklin, J. (1987) Hypertext: An Introduction and Survey. IEEE Computer, September.
9. Davis, H., Lewis, A., Rizk, A. (1996) OHP: A Draft Proposal for a Standard Open Hypermedia Protocol. In 2nd Workshop on Open Hypermedia Systems, Washington DC: University of California, Irvine, pp. 27-53.
10. Davis, H.C., Knight, S., Hall, W. (1994) Light Hypermedia Link Services: A Study of Third Party Integration. In European Congerence on Hypermedia Technology (ECHT '94). Edinburgh, UK. ACM.
11. Fischer, G., McCall, R. (1989) JANUS: Integrating Hypertext with a Knowledge-based Design Environment. In Proceedings of Hypertext '89, Pages 105-117.
12. Fox, E. A., Akscyn, R. M., Furuta, R. K., and Leggett, J. J. (1995) Introduction to special issues on digital libraries. Communications of the ACM, 38, 4, April.
13. Grønbæk, K., Hem, A.H., Madsen, O.L., Sloth, L., (1994) Designing Dexter-based and Cooperative Hypermedia Systems. Communications of the ACM, 37(2). February.
14. Grønbæk, K., Bouvin, N. O., Sloth, L. (1997) Designing Dexter-based hypermedia services for the World Wide Web. In HYPERTEXT '97 - Eight ACM Conference on Hypertext. Southampton, UK, April 6-11: ACM.
15. Grønbæk, K., Sloth, L., Ørbæk, P. (1999) Webvise: Browser and Proxy Support for Open Hypermedia Structuring Mechanisms on the WWW. To Appear in Proceedings of the Eighth World Wide Web Conference 1999, Toronto, Canada.
16. Grønbæk, K., Trigg, R.H. (1999) From Web to Workplace : Designing Open Hypermedia Systems, MIT Press.
17. Halasz, F.G., Moran, T.P., Trigg, R.H. (1987) NoteCards in a Nutshell. CHI/GI 1987 conference proceedings on Human factors in computing systems and graphics interface.

18. Halasz, F., Schwartz, M. (1994) The Dexter Hypertext Reference Model. *Communications of the ACM*, 37(2), pp. 30-39.
19. Hansen, K.M., Thomsen, M.: Application Arbitrator and Domain Concealer Patterns - Addressing Architectural Uncertainty in Interactive Systems. In *Proceedings .of TOOLS Asia'99*.
20. Hitchcock, S., Carr, L., Harris, S., Hey, J. M. N., and Hall, W. (1997) Citation linking: improving access to online journals. *DL '97. Proceedings of the 2nd ACM international conference on Digital libraries*, July 23-26, Philadelphia, PA. Pages 115-122
21. HTML (1998) HTML 4.0 Specification. <http://www.w3.org/TR/REC-html40/>.
22. Kacmar, C. (Ed.) (1995) Special Issue on Digital Libraries. *SIGLINK Newsletter*, 4, 2, (September), ACM Press.
23. Lerner, B.S., Habermann, A.N. (1990) Beyond Schema Evolution to Database Reorganization. In *Proceedings of ECOOP/OPSLA '90 Proceedings*. ACM Press.
24. Madsen, O.L., Møller-Pedersen, B., Nygaard, K. (1993) *Object-Oriented Programming in the BETA Programming Language*. Addison-Wesley.
25. Marshall, C.C. (1997) Annotation: from paper books to the digital library. *DL '97. Proceedings of the 2nd ACM international conference on Digital libraries*, July 23-26, Philadelphia, PA. Pages 131-140
26. Maurer, H. (1996) *Hyperwave. The Next Generation Web Solution*. Addison-Wesley.
27. Nanard, J., Nanard, M. (1991) Using Structured Types to Incorporate Knowledge in Hypertext. In *Proceedings of Hypertext '91*.
28. Nanard, J., Nanard, M. (1993) Should Anchors Be Typed Too? An Experiment with MacWeb. In *Proceedings of Hypertext '93*, Pages 51-62.
29. Nielsen, J. (1998) Using Link Titles to Help Users Predict Where They Are Going. Jakob Nielsen's Alertbox for January 11, 1998, <http://www.useit.com/alertbox/980111.html>.
30. Nürnberg, P.J., Wiil, U.K., Leggett, J.J. (1998) Structuring Facilities in Digital Libraries. In *Proceedings of ECDL '98, (Heraklion, Crete, Greece, September)*. Springer Verlag.
31. Ousterhout, J.K. (1994) *Tcl and the Tk Toolkit*. Addison-Wesley.
32. Raggett, D. (1994) A Review of the HTML+ Document Format. In *Proceedings of WWW1. The standard proposal may be found at* http://www.w3.org/MarkUp/HTMLPlus/htmlplus_1.html.
33. Rizk, A., Sauter, L. (1992). Multicard: An Open Hypermedia System. In *European Conference on Hypertext (ECHT '92)*, (pp. 4-10). Milano, Italy: ACM.
34. Röscheisen, M., Mogensen, C., Winograd, T. (1994). *Shared Web Annotations as a Platform for Third-Party Value-Added Information Providers: Architecture, Protocols, and Usage Examples*. Technical Report. Stanford University, Computer Science Dept.
35. Sheridan, P.B. (1959) The Arithmetic Translator-Compiler of the IBM Fortran Automatic Coding System, *Comm. ACM* 2:2 (pp. 149-157).
36. Smith, T. (1995) Report of the Multimedia Perspective Working Group. IITA Digital Libraries Workshop, <http://www-diglib.stanford.edu/diglib/pub/reports/iita-dlw/part5.html>.
37. Trigg, R.H. (1983) *A Network-Based Approach to Text Handling for the Online Scientific Community*. Ph.D. thesis, University of Maryland, TR-1346.
38. Wiil, U.K. (1998). Evaluating HyperDisco as an Infrastructure for Digital Libraries. In *Proceedings of the 1998 ACM Symposium on Applied Computing (SAC '98)*, (Atlanta, GA, February). ACM Press.
39. XML (1999) Extensible Markup Language (XML). <http://www.w3.org/XML/>.