

An Open Linking Service Supporting the Authoring of Web Documents

Renato Bulcão Neto,
Claudia Akemi Izeki
Department of Computing
University of São Paulo
13560-970, Brazil

rbulcao,cizeki@icmc.usp.br

Maria da Graça Pimentel,
Renata Pontin Fortes
Department of Computing
University of São Paulo
P.O. Box 668, Brazil

mgp,renata@icmc.usp.br

Khai Nhut Truong
College of Computing
GVU Center, GaTech
Atlanta, GA 30332-0280, USA
khai@cc.gatech.edu

ABSTRACT

Both content driven web authors and application designers may have their attention deviated from their main task when they have to be concerned with the generation of elaborated linking structures. This work aims to demonstrate how a metadata-enhanced web-based open linking service can be exploited towards supporting content driven authors in their tasks. The following results are presented in this paper: (a) the Web Linking Service (WLS), a novel open hypermedia system that stores and exchanges metadata in RDF standard syntax for hypertext structures across the wire and (b) two case studies in which applications offer to their users the ability to create linking structures upon existing contents by making use the WLS service.

Categories and Subject Descriptors

H.5.4 [Hypertext/Hypermedia]: Navigation; I.7.2 [Document and Text Processing]: Document Preparation—*hypertext/hypermedia*

General Terms

Design, Experimentation

Keywords

Web Engineering, Document Engineering, Content driven authoring, Open linking service, RDF-based metadata.

1. INTRODUCTION

The majority of web-based applications have three main attributes that usually differentiate them from more traditional software applications, they are: (a) network intensive, (b) content driven and (c) continuously evolving; these attributes have a profound impact on the way Web Engineering is conducted [22]. The content driven characteristic

regards to the fact that web-based applications are primarily to present hypermedia content to users. As far as that characteristic is concerned, content managers suffer cognitive overhead during authoring because a document stores both linking information and contents. From the document engineering point of view, application designers must support evolving linking structures. Both content managers and application designers are limited by the simple linking model underlying the Web. More elaborated linking models have been demanded, as demonstrated by the definition of the XML Linking Language (XLink) [9] that specifies how to create and describe links among resources.

More elaborated linking models are usually supported by open hypermedia systems. An open hypermedia system (OHS) is a middleware that provides to applications hypermedia linking functionality orthogonally to their storage and display functionalities. In other words, by using an OHS, applications can create links to and from documents without modifying the information itself [6].

The Flag taxonomy of open hypermedia systems [21] classifies OHSs in *hyperbase services* or *linking services*. The former provides both linking and storage support while the latter provides only linking support. Applications can use a *linking service*, such as Microcosm [18], Webvise [13] and Chimera [1], to implement link-related functionalities – such as *new link* and *follow link* – without having to implement a hypertext engine. In this context, a hypertext engine is a software component responsible for the management and storage of hypertext structures such as anchors, links and contexts (acyclic graph of links and nested contexts). Moreover, applications can use a linking service to support hypertext structures that are independent of the documents their hypertext structures refer to. Finally, hypermedia applications can exchange both hypertext structures and document contents by means of accessing information available in *hyperbase services* such as HyperDisco [24].

Therefore, when authoring web applications, both content managers and application designers are able to separate the task of generation of the document contents from the task of supporting or elaborating appropriate linking structures by using an open linking service. The contribution of existing open linking services to that process can be leveraged off if appropriated metadata is associated to the hypertext structures; however, this demands the extension of existing open linking services to explicitly support metadata. The work reported in this paper aims to demonstrate how a metadata-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DocEng'02, November 8–9, 2002, McLean, Virginia, USA.
Copyright 2002 ACM 1-58113-594-7/02/0011 ...\$5.00.

enhanced web-based open linking service can be exploited to support content driven authors and application designers in their tasks.

This paper presents the following results: (a) the Web Linking Service (WLS), a novel OHS that stores and exchanges metadata in RDF (Resource Description Framework) [17] syntax for hypertext structures across the wire and (b) two examples of applications that use WLS. In the first case, the application offers its users the ability to create linking structures upon existing contents: as a result, authors can concentrate on the content they create on the Web since the task of elaborating complex linking structures is supported by the application in a later stage. In the second case, the application focus is on the automatic generation of links: the application uses WLS as a repository in which the links created are maintained.

This paper is organized as follows. In Section 2 we present the WLS service. In Section 3 we discuss the RDF-based metadata modeling for the WLS service. In Section 4 we discuss the use of the WLS service by third-party applications. Finally, Section 5 summarizes the contributions and points to future work.

2. WEB LINKING SERVICE

Linking services provide a powerful solution to support inter-application linking as a mean of integrating distributed heterogeneous applications and data repositories. The Web Linking Service (WLS) is an XML-based open linking service for the Web implemented as an application programming interface so that application developers can reuse and combine the available operations with their own building blocks. Moreover, WLS can be reused in different contexts reducing the authoring effort of XML-centric applications.

2.1 OHS requirements for WLS API

Applications developed and handled by OHSs are composed of a set of documents (created from different applications) and a set of links stored externally to the documents (into linkbases). A detailed research on OHS applications [20] characteristics aimed to identify and describing which requirements the WLS API must provide in order to appropriately handle OHS applications as follows.

1. **Explicit separation between contents and linking structure.** Key characteristic of OHS applications. No information about links should be contained in applications documents. Therefore, documents can be analyzed, processed and handled by the applications that created them [15].
2. **Support for public and private linkbases.** As consequence of the requirement 1, information about links is handled by external linkbases that can be **public** or **private** [15].
3. **Different relationships between documents.** OHS applications can support different types of links [6, 15] such as: (a) **specific links** one source anchor and one destination anchor, (b) **bi-directional links** two-way specific links, (c) **local links** inserted once and reflected automatically at every occurrence of the source anchor in the source document, (d) **global links** similar to local links, but reflected automatically at every occurrence of the source anchor in every document in

the OHS application, and (e) **computed links** allow a text in the source document to be selected and find other documents that contain a high occurrence of the significant words in the selection.

4. **Links in terms of the information semantics.** Links to information with the same semantic meaning should be supported independently of the documents identity holding the information. Such links are called “content-based links” because they connect semantics rather than locations or file identifiers [23].
5. **Distribution issues should be supported.** OHS applications should be distributed (e.g. computation, storage) across a network and hardware platforms [15].
6. **Cooperation issues should be supported.** The contents and structure of an OHS application can be developed in a cooperative way. Although it is not a key requirement of OHSs, the ability to support asynchronous work is a key concept [6].

Based on the described research, we have also noted two kinds of requirements: the necessary requirements (those which should be found in all OHS applications — requirements 1, 2 and 3), and the desirable requirements (those which could be found in applications of particular OHSs — requirements 4, 5 and 6).

In the following subsection we describe the WLS conceptual model in which we relate every class defined in the modeling to the respective requirement for the WLS API development.

2.2 WLS modeling

The WLS conceptual model (Figure 1) is based on both research about open hypermedia systems’ requirements and the OHP (Open Hypermedia Protocol) navigational model¹. OHP is a bi-directional point-to-point protocol by means of OHSs and integrated applications exchange messages for communication. As the OHP navigational model was defined as an open model, it is allowed both to specialize new classes and include new properties to the OHP model.

The core of the WLS functionality is the relationship among the **Anchor**, **EndPoint** and **Link** classes. The **Anchor** class defines an internal location into the document contents or the whole document. **Anchor’s** value is stored into the *expression* property. The **EndPoint** class corresponds to each extremity of a link (each anchor), but with an additional information: its respective direction (source, destination or bi-directional). For that reason an anchor may be shared among several links endpoints. The **Link** class identifies associations between endpoints. According to the relationships among the **Anchor**, **EndPoint** and **Link** classes, WLS conceptual model provides support to multidirectional external linking structures. In other words, all information about anchors, endpoints and links are stored into external linkbases. Therefore, WLS supports the requirements 1, 2 and 3.

The **NodeRef** class corresponds to references to the documents managed by applications integrated to WLS service. The *URL* property stores the physical location of documents on the Web.

¹<http://www.csd1.tamu.edu/ohs/>

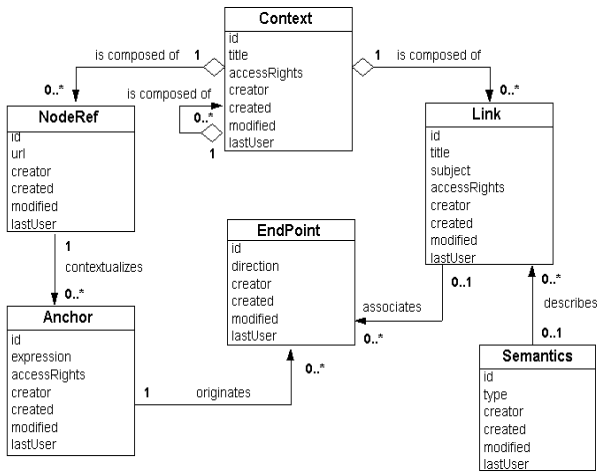


Figure 1: WLS conceptual model: an extension to the OHP model.

The Context class represents a collection of noderefs, links and nested contexts. This class allows both different sets of links over a same set of information and the reuse of previously created navigational contexts. Although WLS uses a single public linkbase to all users, contexts can be seen as private linkbases when combined to access rights, since they can represent individual, group or public hypertext structures over a set of documents. Therefore, WLS also supports the requirement 2.

Since the semantic relationships between links endpoints are not often achievable on the Web and we also aim to support searches based on those semantic relationships, we have extended the OHP navigational model with the Semantics class. For instance, a user may search for documents on the Web not only by means of string matching of their contents but also by means of the semantic type defined by the web author between the documents. WLS provides some pre-defined semantic types for links such as comment, explanation, example, etc. Therefore, WLS may support for the requirement 4.

Regarding to the requirement 5, WLS does not yet support API and linkbase distribution over different hosts on the Internet. So far, applications' processes must be located in the same host where WLS API and its linkbase are.

Metadata are important in collaborative settings to provide awareness about *who*, *which* and *when* hypermedia objects have been accessed and modified. Metadata has been associated to classes in the WLS data model: *creator*, *created*, *modified* and *lastUser*. In order to support private and public hypertext structures, two types of access rights were defined *read* and *write* for Context, Anchor and Link classes. For instance, if a user has no permission to read and write some anchors into a web document, they can not be seen by her. Therefore, by handling awareness information and access rights for users in collaborative settings, WLS supports the requirement 6.

Therefore, the WLS conceptual model extended the OHP model with: (a) metadata for awareness about *who*, *which* and *when* hypermedia objects have been accessed and modified; (b) access rights for private and public hypertext structures in collaborative settings; and (c) the Semantics class for explicit semantic relationships on the Web.

2.3 WLS implementation and architecture

The software platform used in WLS implementation includes the Linux operating system, the Apache web server integrated to a PHP interpreter and the MySQL database. The use of specifications such as XML [3], XPointer (Extensible Pointer Language) [8] and the RDF [17] and RDF Schema [4] is discussed below.

Some OHSs such as Chimera [1] make use of CGI (Common Gateway Interface) for communication. The WLS service uses a PHP API as server-side script language; the set of PHP operations accesses the WLS MySQL linkbase. We have chosen the PHP language because PHP interpreters are available for the most used web servers.

In order to bring the benefits of XML data interchange such as the Webvise OHS [13], we use the XML language as standard interchange format for the communication between the WLS and its integrated applications. Therefore, operations defined for a class of the WLS conceptual model are converted to XML messages.

The WLS service makes use of XPointer as anchoring model for its applications mainly because XPointer can map all the hierarchical tree of XML documents with finer granularity levels.

RDF and RDF Schema were used to associate metadata with objects defined in the WLS data model. Except Webvise, all the web-based OHSs use attribute-value pairs to handle metadata about the stored information. Webvise uses the OHIF (Open Hypermedia Interchange File) for metadata exchange [12], but it is not yet an open hypermedia standard. Therefore, the WLS service is the first OHS that stores and exchanges metadata in RDF standard syntax for hypertext structures across the wire. There is a set of operations from WLS API that extracts metadata for Contexts, Anchors and Links from the MySQL linkbase and returns an RDF string with these metadata.

The WLS API² comprises six sets of operations. The Context API defines methods related with: (a) creation and removal of contexts, (b) user's access control to contexts and (c) generation of RDF-based metadata for contexts. The NodeRef API defines methods related with: (a) creation and removal of nodeRefs and (b) getting the list of anchors of a nodeRef. The Anchor API defines methods related with: (a) creation, removal and updating of anchors and (b) generation of RDF-based metadata for anchors. The Link API defines methods related with: (a) creation, removal and following links, (b) user's access control to links and (c) generation of RDF-based metadata for links. The EndPoint API defines methods related with: (a) creation, removal and updating of endpoints and (b) the definition of incoming and outgoing links from a nodeRef. The Semantics API defines methods related with: (a) creation and removal of semantic types and (b) association between a semantic type and a link.

Although it was not shown in WLS conceptual model in Figure 1, the WLS service uses an API for group and user management as a result of the GroupNode's work, an RDF-based collaborative nodebase service that can be exploited by web applications [16]. Therefore, everything related to user and group management is done by means of that API, such as creation, removal, adding/removing users to/from groups, user authentication, etc.

²<http://coweb.icmc.usp.br/WLSProject/apindex.html>

An open hypermedia system's approach and the Dexter Hypertext Reference Model and its extensions [14] have been used as basis for the design, architecture and implementation of the WLS. According to the three-layer model of hypertext systems proposed by Dexter Model, WLS is responsible for the management of the hypertext network and applications integrated to WLS are responsible for both the storage and presentation of their own documents.

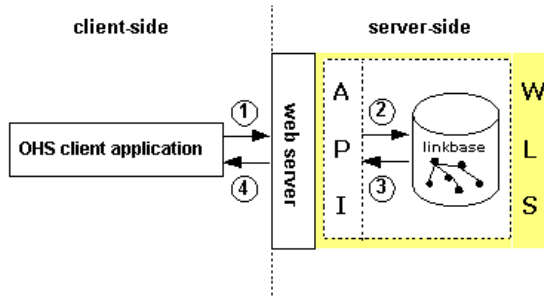


Figure 2: WLS architecture: server-side scripts manage the WLS linkbase for providing linking functionalities to its integrated applications at run-time.

Figure 2 depicts the WLS architecture and its communication with an OHS client application. The process of creation of a link by a client application is as follows:

1. the OHS application must: (a) provide the location of the current document (e.g. a URL), (b) identify and code the anchor selected by the user in the XPointer syntax, (c) assembly the resulting information in an XML message and (d) send the XML message invoking the requested operation from the WLS API;
2. the requested WLS operation accesses the linkbase;
3. the results are returned to the WLS operation in an XML message;
4. the results are sent to the OHS client application. Upon receiving the XML message, the OHS application must: (a) extract the results from that message and (b) provide the hypermedia functionalities by means of its user interface.

The WLS service adopts the mechanism of web integration (see Figure 2) in which the web server is extended with server-side scripts to be a client of an OHS. We chose this integration's approach because it brings the benefit of separating links from web documents in a simple way, since the links are generated and inserted dynamically. This mechanism was also adopted by DLS [18] and DHM/WWW [11] OHSs.

3. RDF MODELING FOR THE WLS

The XML language allows authors to specify the information they handle and interchange in terms of both the structure of the documents and the name of the elements and attributes they contain. The RDF standard, on the other hand, allows the association of semantic information to any resource, such as a web page.

The basic data model of RDF encompasses three basic elements: **Resources**, as anything that has an URI; **Properties**, as specific attributes or relations used to describe a resource; and **Statements**, as associations between a resource and a specific value of its property. For instance, the following Statement "The MIME-type of `http://www.somewhere.net/something#anc` has value XHTML" means that the Resource indicated by URL has a *MIME-type* Property which value is *XHTML*.

A statement can be represented by a direct labeled graph, as illustrated in Figure 3: the resource is represented by an oval node, the value of the property is represented by a square node and the property is represented by an arc connecting the resource to the value of the property.



Figure 3: Graph representation of an RDF model

The RDF basic data model is to be used jointly with RDF Schema in order to allow the interpretation of the semantics declared in the statement, the syntactic definition of statements and the definition of relationships between properties and other resources.

RDF and RDF Schema have been used to provide semantics and metadata for the hypertext structures managed by WLS. The RDF Schema model itself is equivalent to a subset of the class model in UML. Therefore, by means of the WLS conceptual model in UML (see Figure 1) the RDF Schema for the WLS service has been created.

Whilst the Link class in Figure 1 models a link as an object, the following RDF/XML document models a link as a resource. The attributes of the Link class are modeled as properties in the RDF/XML document (along with other properties). We have defined our own vocabulary indicated by "wls:" in which the semantics of such RDF/XML document is specified.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://dublincore.org/2002/08/13/dces#"
  xmlns:n="http://coweb.icmc.usp.br/GroupNode-ns#"
  xmlns:wls="http://coweb.icmc.usp.br/WLSProject-ns#">
  <rdf:Description rdf:about="genid:9857659">
  <rdf:type rdf:resource="http://coweb.icmc.usp.br/WLSProject-ns#Link"/>
  <wls:linkType>biblioenry</wls:linkType>
  <dc:title>Bibliography for WLS</dc:title>
  <dc:subject>
  <rdf:Bag>
    <rdf:_1>WLS</rdf:_1>
    <rdf:_2>linking</rdf:_2>
    <rdf:_3>Master Thesis</rdf:_3>
  </rdf:Bag>
  </dc:subject>
  <dc:creator>rbulcao</dc:creator>
  <n:group>Hypermedia</n:group>
  <wls:created>2001-10-03T21:12:11Z</wls:created>
  <wls:lastUser>otavio</wls:lastUser>
  <wls:lastGroup>HCI</wls:lastGroup>
  <wls:modified>2001-10-03T21:32:35Z</wls:modified>
  <wls:creatorAccess>rw</wls:creatorAccess>
  <wls:groupAccess>r</wls:groupAccess>
  <wls:otherAccess>r</wls:otherAccess>
  <wls:guestAccess>r</wls:guestAccess>
  </rdf:Description>
</rdf:RDF>
```

The following list describes some considerations we have made to define the RDF modeling for that document:

- we have used the `rdf:type` property to indicate that an instance of the resource has all the characteristics that are expected from a class member, i.e., a Link type's instance;
- we have defined `wls:linkType` as an optional property that holds the semantic type of the link;
- we have reused Dublin Core's `dc:title` and `dc:subject` [7] to hold the link's title and to associate keywords to a link, respectively. We have specified the `rdf:Bag` type so that it can hold up to three keywords in `rdf:_1`, `rdf:_2` and `rdf:_3`. We have also reused Dublin Core's `dc:creator` to store the user name who created the link;
- we have defined `wls:lastUser` and `wls:lastGroup` as properties that hold the user who last modified the current link and the group that this user belongs to, respectively. We have also defined `wls:created` and `wls:modified` properties as the dates of creation and last modification of the link, respectively;
- we have reused GroupNode's `n:group` property [16] to hold the group that the creator belongs to;
- we have defined `wls:creator-group-other-guestAccess` as properties with attributes that identify which permissions (in terms of reading and writing) each kind of user and group has.

Since RDF allows the association of semantic information to web resources in XML syntax, its contents can be processed by software tools. Such processing is a basic requirement for the Semantic Web [2], an extension of the current Web wherein applications exchange and process information with well-defined associated semantics.

4. WLS IN USE

4.1 WebNote

In order to demonstrate how an application makes use of the WLS infrastructure, we have extended WebNote³ `pimentel:webnote`, an application that allows a user to have her own web-based repository of annotations independently of any document being annotated.

We have extended WebNote to create links among its annotations since originally WebNote could not support that. We have used the GroupNode service to provide annotations as documents and the WLS service to link annotations to other annotations remotely on the Web.

The basic metaphor of WebNote is a folder in which annotations can be handled collaboratively. Moreover, user can create folders and annotations and give access to other members of her group to read or write the contents of folders as well as the contents of annotations. In order to use the system, a user must be registered on the service and consequently be a member of one group at least.

WebNote makes use of five APIs implemented in GroupNode. In short, this set of APIs (a) handles folders as composites and annotations as specialized nodes, (b) generates

³<http://coweb.icmc.usp.br/webnote/>

XML contents and RDF-based metadata for folders and annotations, (c) allows the annotation visualization as well-formed HTML documents, (d) manages versions of annotations, (e) controls accesses to folders and annotations and (f) (un)locks annotation contents to handle concurrent edition.

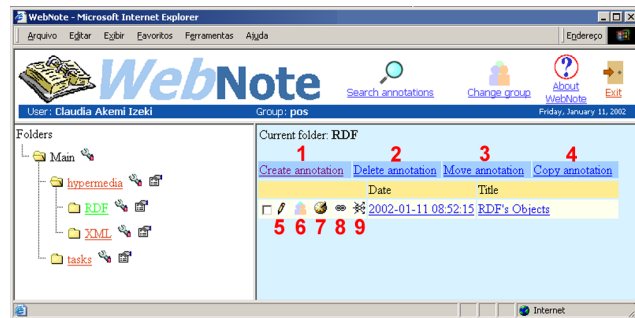


Figure 4: Operations available on annotations: (1) create, (2) delete, (3) move, (4) copy, (5) edit, (6) get/set properties, (7) history, (8) link and (9) show incoming and outgoing links.

Figure 4 presents the operations available on annotations in WebNote. Below the folder's name (**RDF**) in the right-hand frame, hyperlinks allow the activation of the operations (numbered from 1 to 4, respectively) *create*, *delete*, *move* and *copy*. Five icons (numbered from 5 to 9, respectively) are on the left of each annotation: to edit the annotation, to visualize annotation properties and change access rights to annotations, to access the versions of the annotation, to link the current annotation to other annotations, and to show the annotations linked to the current annotation.

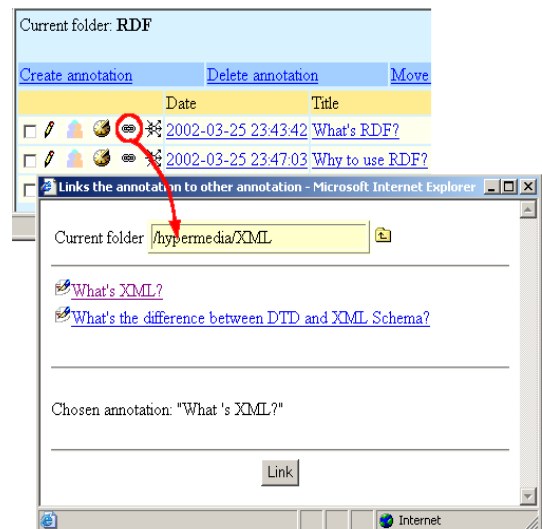


Figure 5: Interface for linking annotations: by clicking on the link icon (background), users can choose the annotation (by means of its title) that they want to link to the current annotation (foreground).

In order to link the current annotation to another one, a user clicks on the icon representing a link (see Figure 5). The list of annotations belonging to the current folder is shown to the user. The user can choose the folder (e.g. *hyper-*

media/XML) in which the annotation she wants to link is. Once chosen the annotation to be linked to the current one (From “*What’s RDF?*” annotation to *What’s XML?* annotation), the user activates the *Link* button (foreground window in Figure 5). The operations *newLink* and *addEndPoint* are requested from WLS API with the URLs of the source and destination annotations as input parameters, respectively.

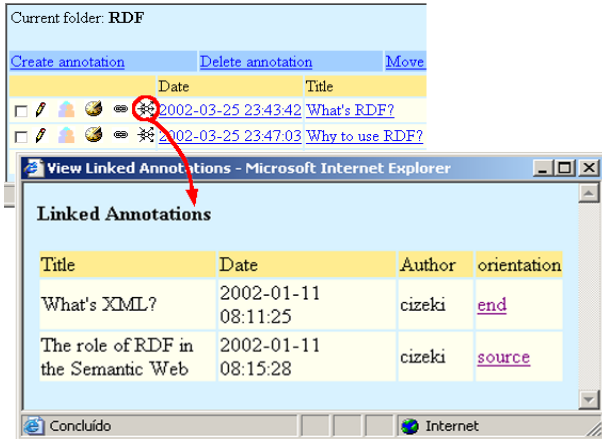


Figure 6: Interface for access of linked annotations to the current annotation: by clicking on the web icon (background), users can get the list of linked annotations with their respective titles, creators and directions (foreground).

If the user wishes to investigate the set of annotations linked to the current one, she clicks on the icon indicated by number 9 in Figure 4). The *linksToNode* operation is requested from WLS API returning to the user the list of the annotations linked to the current one. That list shows not only annotations pointed by the current annotation (indicated by “orientation *end*” in Figure 6), but also annotations that refer to the current one (indicated by “orientation *source*” in Figure 6). The WLS service can provide that information because it stores link’s direction into the WLS linkbase. The user can choose following a link from the list of annotations by clicking on it. The *followLink* operation is requested from WLS API checking if the URL of the annotation selected has been registered into the WLS linkbase. As a result, the user see the destination annotation.

At the time of this writing we have implemented node-to-node links between annotations (whole-component anchors). We have designed the process of following a link when Web-Note is able to address internal anchors into annotations contents as:

1. by clicking on an anchor into annotation contents, the *linksToAnchor* operation from WLS API is requested for returning the list of annotations linked to that anchor. In this case, the list shows the annotations pointed by the current anchor (*end* direction) as well as the annotations wherein anchors refer to the current annotation (*source* direction);
2. by clicking on an annotation’s title, it occurs the normal process of following links as described in Figure 6.

The WLS service aims to overcome the lack of semantic information of the linking model underlying the Web by means of storing and managing RDF-based metadata provided by applications for hypertext structures. Those metadata can be visualized as well-formed HTML by means of XSLT stylesheets [5].

4.2 LinkDigger

LinkDigger⁴ is a service built to identify relationships among pages in homogeneous web-based repositories exploiting information retrieval techniques [19]. Once the relationships are identified, the service makes use of the WLS service to store the relationships as links, which means that the links are stored independently of the source documents.

The top part of Figure 7 illustrates a web-based interface that gives access to LinkDigger: a user specifies two URLs to be processed by the service. After its links generation processing has been concluded, LinkDigger provides a web-based interface giving access to the links identified: this interface, illustrated in the bottom part of Figure 7, is a front-end to the semantic links stored into the WLS linkbase. Semantic links are achieved by exploiting the LSA (Latent Semantic Analysis) technique [10].

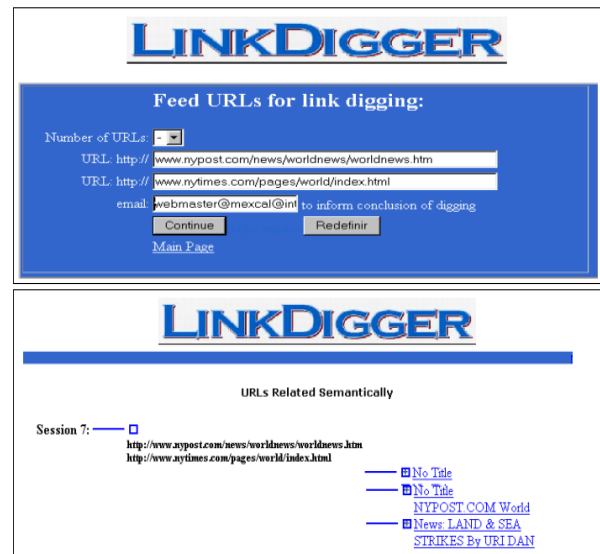


Figure 7: LinkDigger web-based interface: (top) the user provides (up to five) URLs of sites to be linked; (bottom) results of the semantic linking process.

Several experiments have been carried out with LinkDigger. For instance, LinkDigger has related pages from the New York Times (NYT) and New York Post (NYP) online journals. LinkDigger has created 174 latent semantic links among 25 pages of the online journals (16 from NYT and 9 from NYP). More details about experiments with the LinkDigger service can be found in [19].

Since LinkDigger allows the creation of computed semantic links between homogeneous repositories by using WLS infrastructure, WLS also covers the requirements 3 and 4 described in Section 2.1).

⁴<http://mexcal.intermedia.icmc.usp.br/LSL/>

4.3 DocEng Perspective

By means of the WLS service, content managers can elaborate anchors and links transparently upon web documents. In the case of WebNote, link creation is handled **manually** by means of the WebNote's user interface which makes the hypertext authoring a straightforward "pointing and clicking" mechanism.

In the case of LinkDigger, it is responsible for the **automatic** generation of links upon the information contained in web-repositories. In other words, those links are created by the LinkDigger application, not by the user.

Therefore, regarding to applications in general (and in particular WebNote and LinkDigger), all linking information is stored apart from documents contents because links are generated and inserted at run-time. Linking information is always: (a) stored into the WLS linkbase, (b) requested by applications and (c) managed by the WLS API.

5. CONCLUDING REMARKS

We have proposed a metadata-enhanced web-based open linking service which can be exploited towards supporting content driven authors in their tasks. The Web Linking Service (WLS) provides link-related capabilities and collaborative hypertext structures as first-class objects to applications. The core of the WLS service is a network-accessible API that exchanges XML data and RDF-based metadata with those applications.

By means of the WLS service, content managers and application designers can elaborate anchors and links either manually or automatically upon engineering web documents. As a proof of concept we demonstrate how WebNote and LinkDigger applications use the WLS service. The former allows manual link creation by the user and the latter allows automatic generation of links.

Our future plans include support to structure-based searches by using RDF databases in the context of Semantic Web. We also plan to extend WebNote to full support the WLS linking model.

Acknowledgments

The authors would like to thank the Brazilian Funding Agencies (CNPq and FAPESP) and to Carlos R. E. Arruda Jr, Renan G. Cattelan, Aline M. M. Miotto, Alessandra A. Macedo and José A. Camacho-Guerrero for their partial support on this research.

6. REFERENCES

- [1] K. Anderson, R. Taylor, and E. W. Jr. Chimera: hypertext for heterogeneous software development environments. *ACM Transactions on Information Systems*, 18(3):211–245, 2000.
- [2] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. <http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html>, 2001.
- [3] T. Bray, J. Paoli, and C. M. Sperberg McQueen. Extensible Markup Language 1.0 (2nd edition), W3C Recommendation. <http://www.w3.org/TR/2000/REC-xml-20001006>, 2000.
- [4] D. Brickley and R. V. Guha. Resource Vocabulary Description Language 1.0: RDF Schema, W3C Working Draft, 2002. <http://www.w3.org/TR/rdf-schema/>.
- [5] J. Clark. Extensible Style Sheet Language Transformations (XSLT). <http://www.w3.org/TR/xslt/>, 1999.
- [6] H. Davis, A. Lewis, and A. Rizk. OHP: A draft proposal for a standard Open Hypermedia Protocol. In *Proceedings of the ACM Conference on Hypertext and Hypermedia*, pages 27–53, Washington DC, 1996.
- [7] DC. Dublin Core. <http://purl.org/dc/>, July 1999.
- [8] S. DeRose, E. Maler, and R. Daniel. XML Pointer Language (XPointer), Last Call Working Draft. <http://www.w3.org/TR/xptr>, 2001.
- [9] S. DeRose, E. Maler, and D. Orchard. XML Linking Language (XLink), W3C Recommendation. <http://www.w3.org/TR/xlink>, 2001.
- [10] G. W. Furnas, S. Deerwester, S. T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, and K. E. Lochbaum. Information retrieval using a singular value decomposition model of latent semantic structure. In *Proceedings of the 11th International Conference on Research & Development in Information Retrieval*, pages 465–480, 1988.
- [11] K. Grønbaek, N. Bouvin, and L. Sloth. Designing Dexter-based hypermedia services for the World Wide Web. In *Proceedings of the ACM Conference on Hypertext and Hypermedia*, pages 146–156, Southampton, UK, 1997.
- [12] K. Grønbaek, L. Sloth, and N. Bouvin. Open hypermedia as user controlled meta data for the Web. In *Proceedings of the Ninth International World Wide Web Conference*, Amsterdam, The Netherlands, 2000.
- [13] K. Grønbaek, L. Sloth, and P. Orbaek. Webwise: browser and proxy support for open hypermedia structuring mechanisms on the WWW. In *Proceedings of the Eighth International World Wide Web Conference*, pages 253–267, Toronto, Canada, 1999.
- [14] K. Grønbaek and R. Trigg. Design issues for a Dexter-based hypermedia system. *Communications of the ACM*, 37(2):41–49, 1994.
- [15] K. Grønbaek and R. Trigg. *From Web to Workplace: Designing Open Hypermedia Systems (Digital Communication)*. MIT Press, Boston, MA, 1999. 386p.
- [16] C. A. Izeki, R. F. Bulcão Neto, M. G. C. Pimentel, A. M. M. Miotto, and R. P. M. Fortes. A Dual Open Hypermedia Service for the Semantic Web. In *Proceedings of the VIII Brazilian Symposium on Multimedia and Hypermedia Systems*, Fortaleza - CE, Brazil, 2002. To appear in October.
- [17] O. Lassila and R. R. Swick. Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation. <http://www.w3.org/TR/REC-rdf-syntax>, February 1999.
- [18] D. Lowe and W. Hall. *Hypermedia and the Web – An Engineering Approach*. John Wiley & Sons, Chichester, UK, 1999. 626p.

- [19] A. A. Macedo, M. G. C. Pimentel, and J. A. C. Guerrero. An infrastructure for open latent semantic linking. In *Proceedings of the ACM Conference on Hypertext and Hypermedia*, pages 107–116. ACM Press, June 2002.
- [20] A. M. M. Miotto and R. P. M. Fortes. A formal model for applications in open hypermedia systems: specific characteristics. In *Proceedings of XXVII Latin America Conference on Informatics (CLEI'2001)*, Mérida, Venezuela, 2001. In Portuguese.
- [21] K. Osterbye and U. Wiil. The Flag taxonomy of open hypermedia systems. In *Proceedings of the ACM Conference on Hypertext and Hypermedia*, pages 129–139, Washington DC, 1996.
- [22] R. Pressman. *Software Engineering: A Practitioner's Approach*. Mc-Graw Hill, New York, NY, fifth edition, 2000. Chapter 29.
- [23] L. C. Tai. Architecture support for content-based hypermedia. In *Proceedings of the 2nd Workshop on Open Hypermedia Systems*, pages 1–5, Washington DC, 1996. ACM Press.
- [24] U. Wiil and J. Leggett. Workspaces: The HyperDisco approach to Internet distribution. In *Proceedings of the ACM Conference on Hypertext and Hypermedia*, pages 13–23. ACM Press, 1997.