A LIGHTWEIGHT APPROACH TO UNBREAKABLE LINKS IN WWW-BASED HYPERTEXT ENVIRONMENTS

"Users and tools want to break links"

Thomas Bopp, Thorsten Hampel, Bernd Eßmann

University of Paderborn, Computer Science

Keywords: Link Consistency, hypertext, CSCL.

Abstract: In this paper, we present a lightweight approach to achieve link consistency through a combination of object pointers and WWW-style path-oriented links. Our goal is to allow the use of common web-based tools with our CSCW/L system steam, but at the same time achieve link consistency within the system.

1 INTRODUCTION

The WWW has emerged as the de facto presentation standard of all hypertext systems. Therefore most of the Hypertext, CSCW and CSCL systems provide standard web-based user interfaces.

Regarding links, users and tools are "socialized" in a path-oriented thinking. The HREF-style of expressing a hyperlink as a path to a file is an easy to understand and common way to interlink documents. This simplicity may be one of the key features of the WWW and it's success.

A page on the WWW does not only consist of a single document. There are lots of linked documents and images in each hypertext document. Maintaining those structures is not an easy task and therefore hypertext systems provide tools to support the user.

Nevertheless with "normal" web technology no link consistency is achieved. If a document is deleted, renamed or moved to a different location hyperlinks within different documents pointing to that particular document are no longer valid – Link consistency is therefore an integral part of modern hypertext systems.

On the other hand, hypertext systems providing hyperlink consistency use an ID-oriented style and are therefore restricted to the hypertext system's tools and methods of replacing hyperlinks. This leads to proprietary solutions- widely used tools are often not compatible to these hypertext systems.

We distinguish between two types of links: local links and external links. Without a common standard it is impossible to achieve link consistency for external link, as long as the software used at the locations is different.

Local links are usually relative or absolute filenames. The first have the advantage of being independent from the location on the server. Due to that the documents might be moved into a different location.

The approach presented in this paper and implemented in our open source CSCW/L web-based system sTeam (Hampel and Bopp, 2003) (structuring information in teams) allows most tools and "WWW-socialized users" to create links in the common way, but at the same time achieves link consistency within the system. The links are just loosely coupled and the link consistency is only checked when retrieving the document.

This consistency checking method is a replacement of the original method, which used fixed pointers to the documents. There has definitely been an urge to replace the old method because it caused a lot of confusion for the users. The idea of our new concept is to focus on the main goal of link integrity, which is to avoid "Document not found" results. Thus our approach becomes active when the link is not locatable by the given path name. At that point the stored pointer is used to locate the object and the link is replaced by a working one.

Obviously this method only checks local link that point to a location on the same server than the source document. The only solution to this problem is additional standards, which are supported by the browsers.

Bopp T., Hampel T. and Eßmann B. (2005).
A LIGHTWEIGHT APPROACH TO UNBREAKABLE LINKS IN WWW-BASED HYPERTEXT ENVIRONMENTS - "Users and tools want to break links".
In *Proceedings of the Seventh International Conference on Enterprise Information Systems*, pages 251-255
DOI: 10.5220/0002550502510255

Before we describe the handling of links in the sTeam, we first briefly describe the basic ideas behind our system.

1.1 The sTeam system

The sTeam system combines the idea of a room-based virtual world with the basic functionalities of document management. Rooms are social meeting places and centres of a virtual learning community. as a basis for cooperative learning. In a virtual room, it is possible to observe which cooperative partner is currently dealing with documents and knowledge sources.

A room can be viewed by a web browser. It displays all the documents or for example a single hypertext document. Other applications like a whiteboard will view the same room in a different way. Every single document will be placed in a two dimensional position.

In the next section we describe how hypertext documents are handled by our system.

1.2 Current Situation – Link Consistency in sTeam

The sTeam CSCW System uses object pointers instead of pathnames in most situations to guarantee consistency regardless of object positions within the system. The advantage of this method is that even though some document has been moved, the reference would still point to the correct object. This approach prevents links from being broken as described by Nentwich (Nentwich et al, 2003). On the other side this behaviour has one important drawback: Users are used to pathnames and folder structures since it is the common structure used by all of the operating systems. Furthermore pathnames are used as part of URL by the browser and they provide a better readable addressing scheme than object ids.

If a user moves one document and puts another document with the same name in its location he/she would think that a reference would point to the new document. Because the sTeam system replaces all links by object pointers the reference would still point to the original object.

Apparently the object pointer method leads to some misunderstandings, especially when users deleted a document to replace it with a new one. Since a deletion command does not remove the document completely and moves it to a trash bin to provide undelete functionality, some links point to deleted documents. The user is confused about the unchanged document.

There are also clients using movement of documents to keep backups of the files (Apple's WebDAV implementation, Konqueror's FTP mode). The original document is moved and the new one is uploaded with the old name. This breaks any link consistency, because the original shouldn't be moved, but its content updated. Obviously those tools expect a path-oriented server and are using that behaviour to achieve backup functionality.

2 RELATED WORK

Link consistency is found in most hypertext systems such as Xanadu (Nelson, 1980) or the HyperG/Hyperwave system (Kappe, 1995). The latter one is rather interesting, as it converts WWW-path-style links to ID-based links when a document is put into the system. Nevertheless problems occur when documents are edited/exchanged with a common tool.

Modern XML based tools provide links in an external document using W3C standards like XLink (Xlink, 2001) and XPath. Xlinkit provides a lightweight framework using these technologies. This tools still use a file-based approach.

Ingham (Ingham et al, 1995) describes an object model for the World Wide Web. In this model published resources are objects instead of files. The system architecture takes advantage of object-orientation and objects are able to communicate with each other. Referential integrity is provided by bind- and unbind- operations if resources are linked. Apart from that the number of references is counted.

Nentwich (Nentwich et al, 2003) explained a mechanism where links could be repaired in time. This is quite similar to our approach, but it uses a search mechanism to find the data instead of the direct lookup of the unique ID of an object within our system. Also, instead of using the hypertext viewer to display the document, our method is completely server based.

The popular Wiki system achieves link consistency through self-defined path oriented syntax to specify links within the system. Since it does not provide deletion nor movement functionality the Link consistency is provided anyway. Documents are written in the Wiki format, which is converted to HTML before it can be displayed in a browser.

3 HTML LINKS

In HTML there are a couple of tags providing links to other documents. For example there are <script>, <img> and most importantly the <a> tags. All of those tags use a path syntax to specify the link location. This basically means a file system is expected and no object-oriented database. Since object-oriented systems work with unique object identifiers this causes some incompatibility issues. Of course it is possible to map a file path syntax to the object space, but the unique identification is not always possible.

It is advantageous to have unbreakable links for any of the HTML-tags described above, because not only the html documents could be moved, but also images and javascript code used by an html page. The link itself is an attribute of the tag in the form *<tag attribute=value>* and usually is called *href* or *src*

.In some cases the links are obfuscated though when for example Javascript is used to create a link. In that case a function is activated when the user clicks on a link (there is a *onClick* attribute). The function then opens the new location and there is no way to detect these links. Due to that our implementation focuses on the detectable links. Unfortunately some HTML-editors produce complicated HTML output with links, which cannot be detected.

In summarization we have to be aware of the World Wide Web being a path-oriented system. URL (Uniform Resource Locator) is a path-oriented syntax for expressing a location on the Internet.

4 LINK CONSISTENCY, BUT USER- AND WWW-COMPATIBILITY

Up to now, exchanging the links of a document to pointers has provided the consistency of hypertext links in the sTeam system. Each of the links was replaced by an unbreakable link using the ID of the referenced object. Unfortunately this leads to a lot of problems regarding the editing of objects, because when the user edits a document his/her links are immediately exchanged by a cryptic path name.

A solution to this problem is to re-exchange the links again, so the Object identifiers are only used internally. Every time the hypertext document is displayed the ids are replaced again with appropriate filenames. The user is not confused by the internal format and the documents can be saved on the client machine. All the links need to use relative pathnames in order to work everywhere. The disadvantage of this solution is the additional processing time used for exchanging the links.

Due to the shortcomings mentioned above the link consistency has been updated to use a *validation attribute*. That is the document link structure is untouched and the tags are expanded by one

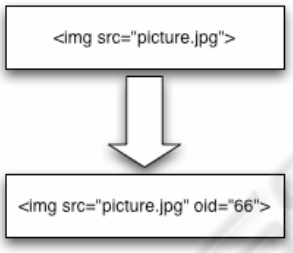

Figure 1: Upload of a hypertext document

additional attribute. When downloading the document all of its local links are validated and possible altered to a new location (external links cannot be checked, because there is no standardized way to figure out the movement of files on other servers). Therefore the links of the document “index.html” in figure 1 would be extended with the known object ID (OID) for “introduction.html” and “overview.html” (#33 and #42). The external link to www.somedomain.org remains untouched. The validation process first evaluates the *href* attribute of the tag and distinguishes between two different situations:

a) The referenced document is found at the given location. The *object* attribute is checked whether it is still the same object as initially referenced. If the object is changed, then we update the *object* attribute. Most likely the author updated the document and the original link will be broken.

b) No document is found. The *object* attribute is used for retrieving the original document and the link does still work. The *href* attribute is attributed to the new location of the document in order to make the document structure still valid.

This method has some advantages: the source code of a document remains almost unchanged with the links untouched and still readable. Only if no document is found the link is replaced by using the provided OID. Apart from that the no re-exchanging of links is done. The additional attribute should not cause too much confusion for the user and is ignored by other applications.

Of course there is still the possibility of “Document not found” results, because if a linked document is just deleted and gone, there is no way to find an appropriate replacement. In this case it makes sense just to remove the link and thus show

the user directly in the source Document that there is a broken link.

4.1 Link Parsing

Technically the parsing is not an additional step in the hypertext processing, because the sTeam CSCW system already supports a feature called RXML or Roxen XML (Nilson, 2001). This extends the html syntax by special tags, which are parsed and replaced in this processing step. The result is additional html output. This feature can be used to create components, which can be reused in several places. Examples of such components are calendars or news-tickers.

Furthermore the normal HTML link tags are also seen as RXML tags and processed in the same way. Each tag has an associated handler script, which handles the replacement of the html code. Thus the parsing is done the normal way with all the link tags registered to the parser. When a link tag is found (for example <img src=”picture.jpg”>) then the appropriate handler script is called which will check if there is a picture.jpg in the current path (when uploading the document) and the tag will be extended by the OID of the picture as shown in figure 1.

When downloading the file there will be a parsing process, which will check the image tag with another handler script. The script checks if the picture is available and if it does not find it, then the link will be updated to contain a path to the new location of the image. Figure 2 shows the result of this process: The image has been moved to the *pictures* sub folder, but still is found and the image *src* attribute is changed to the new location.

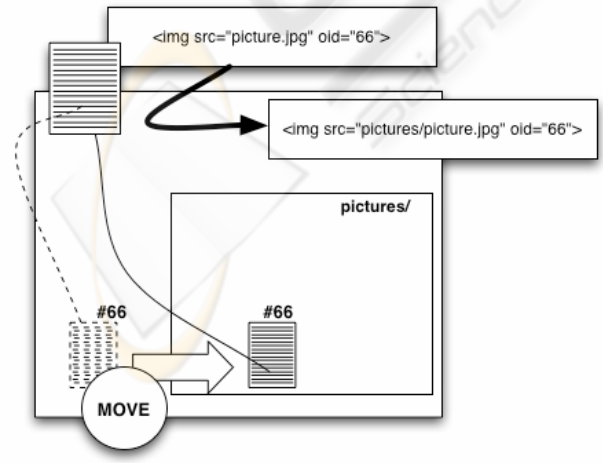

Figure 2: Download and parsing of a hypertext document.

The parser is based on the libxml2 (LibXML2, 2004) SAX parser. The performance of libxml2 is sufficient for hypertext documents and gives only little overhead (documents are usually small and easy to parse).

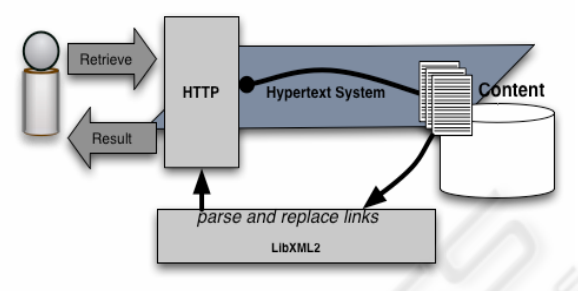

Figure 3: HTML Content Retrieval in the sTeam system.

Figure 3 shows how a user gets a document from the Hypertext system. The http module takes the request and retrieves the content from the document associated with the URL. Before the content is returned it is processed with the libxml2 parser and links are validated and possible exchanged. The result of this process is illustrated in figure 1 and 2.

5 CONCLUSION

Our experiences with the sTeam system during the last years show that it is very important to adapt to the style users are used to work with document structures. Replacing a document through a newer version belongs to the most common tasks. It is crucial to avoid confusion of the user and be as compatible as possible with other tools.

It is also crucial to support web-based interfaces so the hypertext documents can be displayed in a normal browser.

The approach presented in this paper provides link consistency, but at the same time adopts the way and syntax users and their web-based tools are used to create links. It is therefore a good compromise between full link consistency through the logical separation of links and documents, but on the other hand provide the easiness and simplicity of path-oriented embedded links such as the WWW-syntax.

REFERENCES

Davis, H.: *Hypertext Link Integrity*. ACM Computer Surveys. Volume 31, Issue 4, Article No. 28, 1999.

Hampel T.,Bopp T. (2003): Combining Web Based Document Management and Event-Based Systems - Integrating MUDS and MOOS Together with DMS to Form a Cooperative Knowledge Space. ICEIS 2003, Proceedings of the 5th International Conference on Enterprise Information Systems, pages 218-223.

Ingham, D.; Little, M.; Caughey, J.; Shrivastava; S.: W3objects: Bringing Object-Oriented Technology to the Web. In Proc. Fourth International World-Wide Web Conference, 1995, 89–105.

Kappe, F. *A Scalable Architecture for Maintaining Referential Integrity*. Distributed Information Systems. J.UCS 1(2), February 1995.

LibXML2: Homepage: http://www.xmlsoft.org/, 2004.

Nelson, T.H. *Replacing the Printed Word: A Complete Literary System*. Lavington, S.H. (Hrsg.): Information Processing 80. Amsterdam: Publishing Company, 1980, 1013–1023.

Nilson, Martin. The soul of RXML. http://community.roxen.com/articles/011_tag_1/, 2001.

Nentwich, C.; Emmerich, W.; Finkelstein, A.; Ellmer, E. *Flexible consistency checking.* ACM Transactions on Software and Methodology. Volume 12, Issue 1, 2003.

XLink: W3C Recommendation 27. June 2001: XML Linking Language. http://www.w3.org/TR/2001/REC-xlink-20010627/