Advanced XHTML Plug-in for iServer

Semester work

Stefan Malaer
<stefan@malcom.ch>

Prof. Dr. Moira C. Norrie
Dr. Beat Signer

Global Information Systems Group
Institute of Information Systems
Department of Computer Science

12th October 2005

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

globis

Abstract

The iServer architecture is an extensible cross-media information platform enabling links between arbitrary typed objects. It provides some fundamental link concepts and is based on a plug-in mechanism to support various media types. The goal of this semester work was to develop a XHTML plug-in for iServer which enables links from XHTML documents to other XHTML documents as well as parts of them. The resulting iServext is a Firefox extension for iServer which provides visualization and authoring functionality for XHTML links. Furthermore, we investigated research in the area of link augmentation and provide an overview of recent technologies.

Contents

1 Introduction

Information becomes an increasingly valuable resource in today's world. Not only information storage but also information access and crosslinking are in the focus of interest. The World Wide Web is the largest collection of information and therefore of particular interest. Of course, information can also be stored in the form of other media types. Various libraries and archives hold an enormous amount of information. The idea of linking information together is much older than the Web and was for example applied in literature citations. Through the Web the so called hyperlinking has become a central concept of an information management.

The improvement of the current Web linking model was a topic of many research projects. Most of them intended to improve the functionality of the current links with better information about the link target, external link databases, easier authoring and others features. A survey of the various project can be found in Section 2.2. In addition there was already a project for implementing an XHTML plug-in for iServer based on a proxy approach [35].

All these projects, except the one mentioned at the end, only focus on linking within the Web. Most of them address one specific topic of link augmentation. A more general approach to link information is needed where not only resources on the Web can be linked but also resources of other media types can be addressed. The existing XHTML plug-in for iServer intends to do this but has some limitations due to the proxy approach.

The iServer architecture is an extensible cross-media information platform enabling links between any kind of digital and even physical resources. The iServer architecture provides fundamental link concepts such as resource addressing, layering and user management. New media types can be added based on a plug-in mechanism. The main goal of this project is to develop a plug-in for the iServer framework enabling link from and to XHTML documents and parts of them.

We have implemented an XHTML plug-in architecture based on a Firefox extension and the iServer Web Service [15]. Instead of inserting the link information into a document on the server side, this is done within the Web browser which also does the visualization of the links. The link information is stored in an external database and the authoring of new links can be done independently of the accessibility of the document for alerting. In this report we describe the theoretical background of link augmentation and the details of the implementation.

The document is organized as follows. We discuss various topics of augmented linking and related work in Chapter 2. The link visualization and the XLink standard are also addressed in this chapter. The iServer model and its role in the context of link augmentation is presented in Chapter 3. The architecture and the implementation of the XHTML plug-in are described in Chapter 4 and 5, respectively. In Chapter 6 we draw conclusions from our work, outline current limitations and discuss possible future work. In Appendix A the development environment for Firefox extensions is described whereas in Appendix B a user manual is provided. Appendix C lists some references for potential future work.

2 Augmented Linking

In the following sections, we explain the need for augmented linking, present related projects, link integration and authoring. Further, we address link visualization and XLink as a possible approach.

Augmented linking includes all forms of enhancement to the current linking model of the World Wide Web. This is one method for a general Web augmentation which is being investigated intensively.

2.1 Need for augmented linking

Although one of the biggest strengths of the Web is the simplicity of the link model, this is also a significant shortcoming. With the ongoing evolution and maturing of the Web this simplicity needs to be addressed.

In this section we will discuss the current link model with its strengths and limitations. Afterwards, we will give two possible approaches for link augmentation.

2.1.1 Current link model

The current link model supports HTML anchor tags `<a>` only, which offer a set of attributes for defining link properties[1]. The most important one are the `href` attribute which identifies the `target` URI and the target attribute which defines how the linked document will be handled, e.g. opened in the current or a new window. There are many other attributes like title, rev (reverse link) and rel (relationship to linked target), but they are rarely used. A special form of a link are the `<object>` tag and the `<img>` tag . They are used to include the target source (e.g. an image or a flash film) directly in the document.

[1]Further information: http://www.w3.org/TR/html4/struct/links.html

Strengths

As mentioned earlier the main strength of the linking model is its simplicity . It is very easy to understand and implement in existing documents. The same anchor tag is used for navigational and contextual links. Although a separation may be advisable, it makes using the model easier. It is also possible to specify links within a document or in another frame. The linking to other types of media is supported too, but only to entire files.

Limitations

Future developments with more flexible ways of linking and different types of media are restricted by the current model. Some of the limitations as described in [43, p.11] and [6, p.39] are:

- Embedded links allow only the author and owner to define links
- Embedded anchors limit the ability to link from or to read-only material (e.g. CD-ROMs, Films, PDFs)
- Embedded unidirectional links make backtracking extremely difficult
- The lack of overlapping anchors creates problems
- Links lack any form of inherent contextualization
- Single-source, single-destination links inhibit the representation of certain types of associations
- Untyped links inhibit intelligent browsing and navigation
- Embedded and untyped links restrict effective content management and flexible information structuring schemes
- Only one set of links per document

A main problem is that links are embedded in the documents. The Web link is little more than a goto or a jump instruction to the Web browser to retrieve or display a new document. As links are unidirectional, there is no way to determine whether there are links pointing to a given document. There is also no certainty whatsoever that the destination of a link is still available, what can make browsing very annoying. The single-destination links and the lack of multiple sets of links limit the possibility for a user to choose what kind of information in terms of linking he likes to have displayed. Also the creation of temporary link sets for destinations which are only temporary available is not possible. This does not mean that the Web link model is insufficient in general, but for some purposes an augmentation of the link model would be appropriate.

2.1.2 Approaches for link augmentation

One approach are innovative solutions using existing technologies. Shortcomings like single-source, single-destination links and non intelligent browsing can be eliminated. For example to achieve a richer link model the scripting language JavaScript is used in [6, p.18]. With the help of a small script multi-source and multi-destination links can be supported. The script provides a linking function, which can be called from different anchors and pass the name of the link whenever they are activated. This means multi-source anchors can trigger the same link destination. The function has also the possibility to open more than one link at once and therefore provides multi-destination links. Additional JavaScript code can be added to allow for example attributes which change the navigation behavior based on user configuration or other characteristics.

Another important approach is to store links outside of the documents, in an external link database. This approach relies on the use of new technologies and has to be implemented additionally in existing architectures. There are numerous ways available for the storage itself. Since the Web does not support easy integration of new features, an interface has to be developed which enables the integration of external linkbases into it. Some examples of such interfaces are discussed in the following section.

2.2 Related work

There have been many projects and studies dealing with Web augmentation, especially in the area of editing, retrieval and visualization of links. In this section an overview of existing work is given according to [5].

2.2.1 Chimera

This systems was used to carry out experiments with either displaying structure information in a separate programm or making the structure server accessible through HTTP [1, 2]. A Web server was modified to interpret an HTTP request as a request to a Chimera server to which the Web server was hooked up and in turn translate the Chimera structures to HTML. In this mode a user was able to browse the hypermedia structure using an ordinary browser. At the other hand, a Java applet capable of displaying Chimera hypermedia structures was created. By the combination of s special Web server, CGI-scripts and cookies, this applet was inserted into all pages displayed in the Web browser, giving the user immediate access to Chimera services.

2.2.2 Hyper-G/Hyperwave

This system relies on a custom document format (OHF) as well as Web servers and clients (Harmony) [30]. The hypermedia system offers a strong support for hierarchical structures and searching and allows users without a special browser to create links using forms. It is also possible to interface to the system using an ordinary Web browser using a special WWW-Gateway that translates OTF documents and hypermedia structure to HTML. The latest versions is known as Hyperwave[2] and offers a wide variety of services.

2.2.3 Distributed Link Service

This approach uses a wrapper to attach a simple client interface consisting of menus in the browser's toolbar for the creation of new links [9]. The systems is based on the Microcosm hypermedia system [10, 8, 26]. Links are followed by selecting 'Follow Link' in a contextual menu. A link server was then requested and sent a Web page of the matching links. As an improvement an interfaceless version was developed that used a link server proxy to modify the pages on the fly by inserting links in Web pages as requested by the user. The user is able to configure which link bases should be used. Due to performance issues and copyright concerns about adding content to Web pages, a new design was introduced in the form of AgentDLS [7]. Links are now displayed in a separate window and become more of a advisory nature.

2.2.4 DHM/WWW and Extend Work

The University of Aarhus, Denmark, worked on various projects. They created DHM/WWW which was enhanced by Navette and Webvise. Later, the Arakne Framework was developed to provide a generalized Web augmentation tool.

[2]http://www.hyperwave.com

DHM/WWW

The DHM/WWW was an attempt to enhance Web linking with an architecture consisting of a Java applet communication through a CGI-script to a DHM Server [21]. When the user requests a document in the applet, it retrieves the document while querying the DHM server for endpoints in the document. The endpoints are downloaded and displayed in the browser window based on JavaScript functionality.

Navette

DHM/WWW was enhanced by Navette which applied a signed Java applet, to use Web pages from arbitrary Web servers [6]. Navette was embedded in a frame in the Web browser, as the small DHM/WWW window tended to disappear under other windows. Link decoration was handled through a proxy thread as the Netscape browser supported dynamic proxy configuration changes through digitally signed JavaScript. Navette supports multiple users and collections of links.

Webvise

The Webvise client [23, 22] was developed simultaneously with Navette. It is a custom integration with Microsoft Internet Explorer. Webvise could insert links after the browser had displayed the document, thereby improving the overall performance. This is done through DOM and the COM interface available in the Internet Explorer. The COM interface is a plug-in mechanism which gives full control of the browser but ist limited to this specific Web client only.

Arakne Framework

As a follow-up project the Arakne Framework is an object-oriented, component-based model aimed at providing Web augmentation tools a unified access to structure servers, proxies and Web browsers [6, 5]. The system is split into three layers, which are a content layer, a service layer for handling navigation, integration etc. and a structure layer for storage and retrieval of structure. It can be seen as a framework generalizing Web augmentation tools integration.

2.2.5 HyperScout

This project investigates whether navigation can be supported by providing preview information about the links and their targets to the user [41, 42]. The goal is to determine which information should be provided to the user and how it can be displayed efficiently. The preview information was about the title, description, response time, language, last update, last visited, size of the document, type of the link and the action of the browser when the link is clicked. Their conclusion was that some additional information can improve the usability of links evidently, but too much information can confuse the user.

2.2.6 Link Visualization with DHTML

This project focuses on the topic of link visualization of extended links [34]. For visualization DHTML was used, which can be displayed in a regular browser. As an example it uses an XLinkbase, from which the links are generated. This technology does not only show additional attributes, but also allows multi-destination links.

2.2.7 Amaya project

The application is jointly developed by W3C and the WAM (Web, Adaptation and Multimedia) project at INRIA. Amaya is a Web editor, that is a tool used to create and update documents directly on the Web. Browsing features are seamlessly integrated with the editing and remote access features in a uniform environment. This follows the original vision of the Web as a space for collaboration and not just a one-way publishing medium [27]. Amaya includes Annotea [38] a collaborative annotation application based on the Resource Description Framework (RDF), XLink, and XPointer.

2.3 Link integration and authoring

The augmented links have to be integrated and authoring capability has to be provided. If no custom built systems are used, this is done through enhancement of existing systems. The possible methods are discussed in this section.

The Web augmentation tools mentioned above are using various methods for integration. The main differences are the point where the integration takes place and how it has been done. There are three different places where the integration is usually done:

- Within the Web server by using a custom Web server, extensions or CGI-scripts on the server. An extended Web server can translate a proprietary data format into HTML (Chimera). With CGI-scripts requested pages can be modified when the are requested.
- Within a local or remote (DLS) proxy server. The pages can be modified transparently to the user. But the performance is limited due to the time required by the proxy to communicate with the link server and to process the documents.
- Within the browser by using a custom browser (Harmony, Amaya), a browser plug-in (Webvise) or a Java-applet (Navette). The Web page can be modified before displaying or after visualization through JavaScript and DOM. An advantage of this method is having no loss of performance due to link integration, because the document can be downloaded and displayed before it is modified.

A combination of these methods is also possible and can extend the functionality of the integration substantially. As mentioned earlier, there are also different ways how the integration can be done. The main difference is whether the Web page itself is modified or another user interface is provided. This can be done by providing additional menus in the browser, separate windows or frames.

The authoring also offers the possibility for variations. As a first element, user interfaces can be provided in at least three different forms:

- A specific browser plug-in providing a link creation interface (DLS)
- An applet either integrated in each page (DHM) or in a separate browser
- A user interface integrated in the displayed document by using ordinary HTML forms (Hyperwave)

Other options like custom authoring tools or XUL-like[3] Web content management integration would also be possible but have not been investigated so far.

A second element is the communication from the authoring tool to the link storage facility. This can be done through HTTP requests with CGI-scripts (DHM/WWW), socket communication from an applet by using the applet signature mechanism (Navette) or by the use of a Web client plug-in enabling unrestricted access to external resources.

[3]XML User Interface Language, further information: http://www.xulplanet.com

A further question which has to be regarded in the context of Web augmentation tools is: Should existing systems be enhanced or completely new systems be created? Both approaches have advantages but also some drawbacks. Therefore, no single answer can be given.

2.4 Link visualization

This section will discuss the topic of link visualization. First, we talk about existing methods. Afterwards possible forms of presentation are introduced and then examples of existing implementations are given. Finally, we provide some notions about on-demand link visualization.

2.4.1 Today's link visualization

There exist defacto standards for link visualization in Web browsers. The most popular browsers like Internet Explorer, Firefox or Opera visualize standard HTML links as underlined text in blue color. Links which have been visited by the browser before are shown in purple color as shown in Figure 2.1. The color and presentation of the links can be altered by applying Cascading Style Sheets (CSS) [39] or by altering the default browser settings.

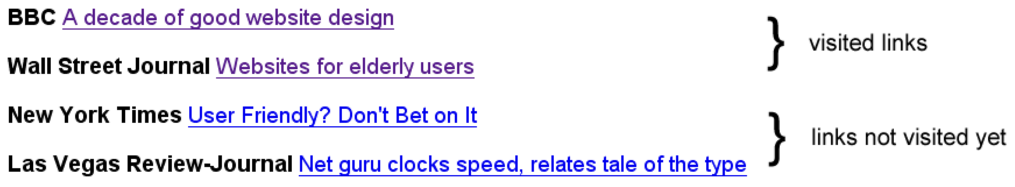

BBC A decade of good website design
Wall Street Journal Websites for elderly users
New York Times User Friendly? Don't Bet on It
Las Vegas Review-Journal Net guru clocks speed, relates tale of the type

Figure 2.1: Standard link visualization

Nielsen suggests to use the title attribute of the `<a href>` tag as an enhancement to display additional information about the link [32]. Nowadays, this attribute is supported by almost any browsers. It can easily be added and is visualized in small window below the link after a second when the mouse cursor is moved over the link. A code sample of a link with a title attribute is shown below and Figure 2.2 shows how it looks when it is displayed.

```
<a href="http://news.bbc.co.uk/" target="_blank"
title="Link news website [opens in a new window]">
```

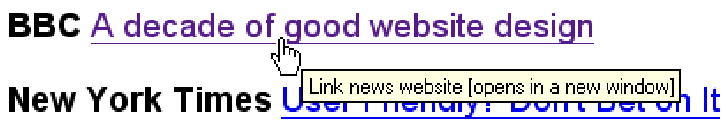

BBC A decade of good website design
Link news website [opens in a new window]
New York Times User Friendly? Don't Bet on It

Figure 2.2: Visualization of the title attribute

2.4.2 Possible presentations of link information

For real link augmentation other possibilities of link information visualization have to be exploited. Problems like multi-destination links, overlapping link markers and bi-directional links have to be addressed. This additional information about links can be displayed in different forms [42]. At least five different forms can be distinguished:

- **Link design:** Different kinds of accentuations for links are a self-evident method to differentiate links. Besides the two standard colors blue and purple, different color densities, background-colors, translucent overlays or font-styles can be used. An advantage is that the coding of link properties are eye-catching. Disadvantages are the reduced readability and restricted information variety the of such methods.

- **At the link:** Additional information can be presented in a short bracketed text or an icon. This method can easily be implemented with today's technology, but can also reduce readability and is not applicable to many layouts and linked graphics.

- **Mouse cursor:** The type of the mouse cursor is another possibility to present the user information about the system status or the action which can be activated by the mouse. Many of today's browsers display a hand symbol when the mouse cursor is moved over a link (see Figure 2.2). The information next to the focus of the user is an advantage. But it is only shown after the mouse is moved over the link and the possibilities are rather abstract.

- **Popup window:** Small popup windows are often used as so called tooltips, they provide for specific user-interface-elements a short additional textual help. They appear if the mouse is kept still for a moment over an element or if an element is clicked. Web browsers also offer right click popup menus where information and options could be implemented. Like the mouse cursor they have the advantage to appear next to the user's focus, but a problem can be that they can hide other elements of the Web site when they are displayed.

- **Other places:** A Web browser offers also other possible places for displaying additional information. The status bar of most browsers can be used and many browsers also offer a side bar. Supplementary content can also be shown in frames, tabs or new windows. This offers a big variety of alternatives, but has the disadvantage that the information is presented off focus.

There are almost no boundaries how additional link information can be displayed. The restrictions mainly come from limited enhancement capabilities of existing tools. Many variants have be implemented and experienced with. In the next section some examples are presented.

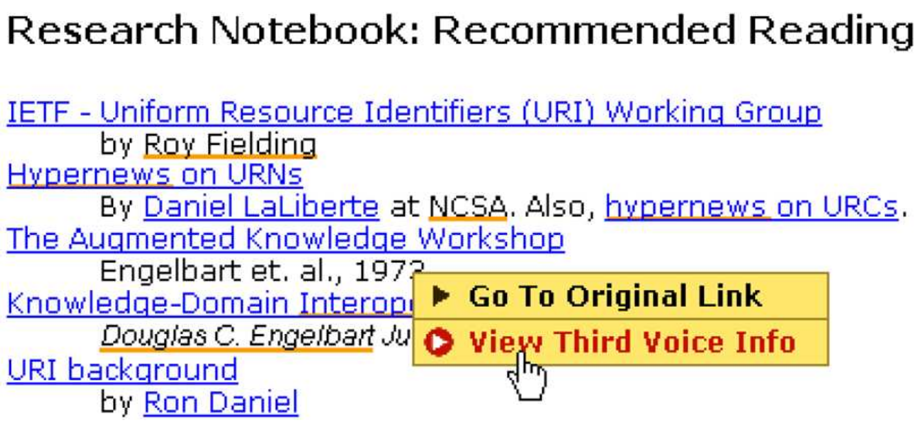
2.4.3 Examples of link visualization

A number of researchers in the area of Web augmentation had ideas about link visualization. Some of them were implemented and others remained just ideas. A selection of the presented concepts are also discussed in [40].

Third voice

An example how additional link markers can be presented and options how to follow them can be presented was given by Third Voice[4]. From an external linkbase additional links were loaded. They were underlined by a thin orange line and could offer a choice of targets in popup windows as shown in Figure 2.3.

[4]The Third Voice project stopped in 2001



Research Notebook: Recommended Reading

IETF - Uniform Resource Identifiers (URI) Working Group
by Roy Fielding
Hypernews on URNs
By Daniel LaLiberte at NCSA. Also, hypernews on URCs.
The Augmented Knowledge Workshop
Engelbart et. al., 197
Knowledge-Domain Interop
Douglas C. Engelbart Ju
URI background
by Ron Daniel

Figure 2.3: Additional links in Third voice

Guide system

The use of different mouse cursors was implemented by the Guide system. The mouse cursor made link characteristics apparent and changed according to the link type if it hovered over a link (see Figure 2.4).

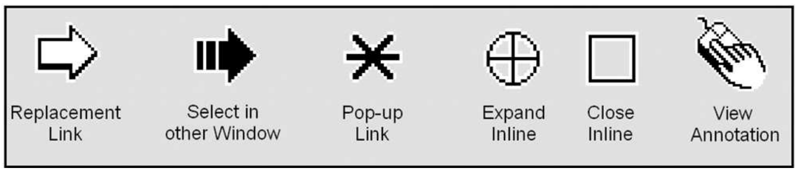
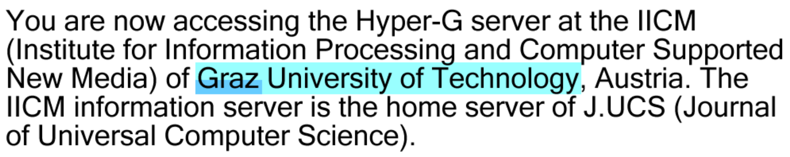


Figure 2.4: Different mouse cursors in the Guide system

Harmony

Harmony, Hyper-G's browser, used overlapping colored background boxes to mark the beginning and end of up to six overlapping markers. For an example with two overlapping links see Figure 2.5. Even these two overlapping boxes decrease the readability of the text and this method will finally fail if a larger number of anchors intersect. The boxes will shrink to pixel height, creating a very distracting background without giving the user applicable information.



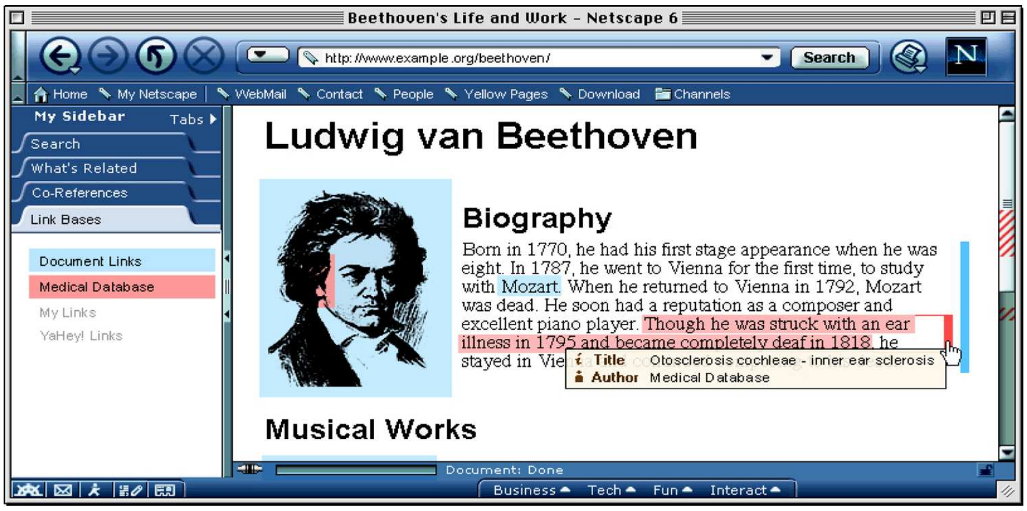
You are now accessing the Hyper-G server at the IICM (Institute for Information Processing and Computer Supported New Media) of Graz University of Technology, Austria. The IICM information server is the home server of J.UCS (Journal of Universal Computer Science).

Figure 2.5: Link overlapping in Harmony

XLink anchors

A possible solution for a high link density is provided by this example of a mock-up of outgoing hypertext links (see Figure 2.6). Therefore a link on-demand technique is used. The different link database may be selected in an additional window or as is done here in the side bar of the browser. By selection or deselection, link databases could be enabled and disabled, allowing the user to view only the links he wants. Colors may be used to associate listed link databases to the anchors on the screen.

Another interesting idea was introduced in this mock-up. When link anchors are longer than a few words, a narrow bar on the right side of the anchored paragraph is used for the marking instead of highlighted text. Good readability can be maintained. The use of the scrollbar is suggested to locate link anchors outside of the current visible section on long Web pages.



Ludwig van Beethoven

Biography

Born in 1770, he had his first stage appearance when he was eight. In 1787, he went to Vienna for the first time, to study with Mozart. When he returned to Vienna in 1792, Mozart was dead. He soon had a reputation as a composer and excellent piano player. Though he was struck with an ear illness in 1795 and became completely deaf in 1818, he stayed in Vie

Title Otosclerosis cochleae - inner ear sclerosis
Author Medical Database

Musical Works

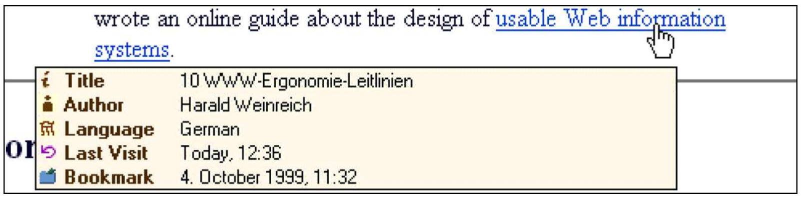
Figure 2.6: Mock-up: Outgoing XLink anchors

HyperScout

In the HyperScout system additional link information is displayed in a pop-up menu that renders XLink-specific and other automatically gathered information (see Figure 2.7). Information can be provided on link target type (”mailto:”, links, downloads), availability (broken links), size and connection speed. Further, attributes of a more semantic nature like title, author and language of the target document or structural hints like indicating out-of-site links could be used to automatically enhance link preview.

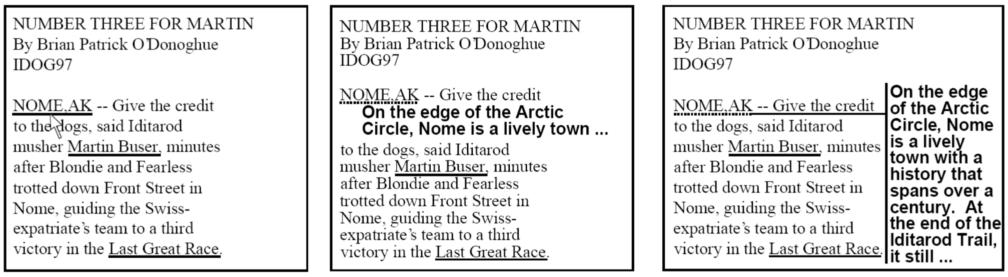
Fluid links

Another approach are fluid links introduced in [44]. They are called fluid because the document adjusts dynamically and makes space for a gloss, which means additional information to a link. This information appears between the lines of text or at the border of the page (see Figure 2.8). An advantage is that the original document is not covered by a pop-up window, but the needed animation is distracting and time-consuming.


wrote an online guide about the design of usable Web information systems.

Title	10 WWW-Ergonomie-Leitlinien
Author	Harald Weinreich
Language	German
Last Visit	Today, 12:36
Bookmark	4. October 1999, 11:32

Figure 2.7: HyperScout


NUMBER THREE FOR MARTIN
By Brian Patrick O'Donoghue
IDOG97

NOME,AK -- Give the credit to the dogs, said Iditarod musher Martin Buser, minutes after Blondie and Fearless trotted down Front Street in Nome, guiding the Swiss-expatriate's team to a third victory in the Last Great Race.

On the edge of the Arctic Circle, Nome is a lively town ...

On the edge of the Arctic Circle, Nome is a lively town with a history that spans over a century. At the end of the Iditarod Trail, it still ...

Figure 2.8: Fluid links

Visual preview

Another method of presenting additional information about the link target is Visual preview [29]. When the mouse cursor is moved over a link anchor either a thumbnail of the link target or a symbol about the link type is displayed next to it (see Figure 2.9). This information can be gathered automatically and does not require any authoring.

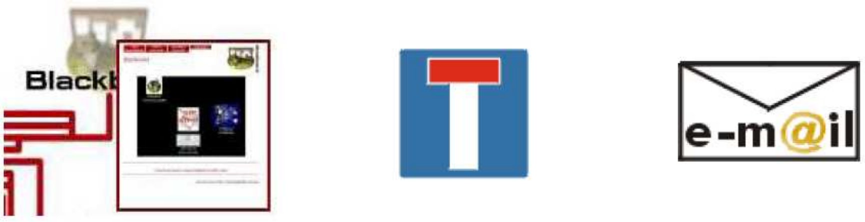
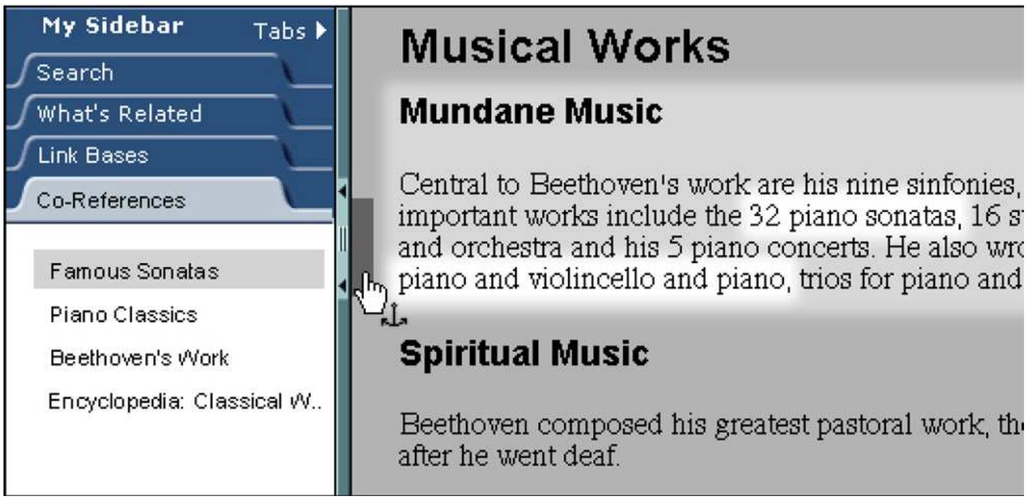


Figure 2.9: Visual preview

Incoming links

In this example no outgoing link markers are addressed but incoming link targets. A possible way of showing them by displaying a lateral maker on the left side. By using the left side they are not mixed with outgoing link markers which can be displayed only on the right side. For a more precise link target visualization, moving the mouse cursor over the maker will shade the rest of the document except for the target area as shown in Figure 2.10.



Musical Works

Mundane Music

Central to Beethoven's work are his nine sinfonies,
important works include the 32 piano sonatas, 16 s
and orchestra and his 5 piano concerts. He also wr
piano and violincello and piano, trios for piano and

Spiritual Music

Beethoven composed his greatest pastoral work, th
after he went deaf.

Figure 2.10: Mock-up: Showing incoming link target

iCab toolbar

Structural links are supported by iCab. They can be browsed with a structural link navigation toolbar (see Figure 2.11). Despite structural information can be provided easily, only very few Web sites offer it.

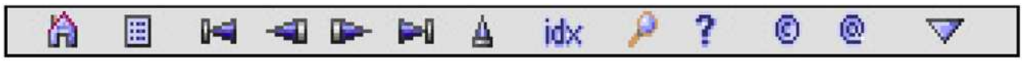


Figure 2.11: iCab's toolbar

We discussed many different methods of link visualization. All of them have advantages but also short-comings. No single best solution can be found and depending on the context best solution possible has to be applied. The idea of links on-demand is further considered in the next section.

2.4.4 Links on-demand

Usually link marker visualization is shown permanently. Another option would be that they are only shown on-demand. For example you have to press a key or click a button to see the links. When you read the text you do not have disturbing link markers, but if you want to browse a text, they can be activated easily.

In [33] two studies are described concerning link marker visualization. The first study compared the standard blue underlined links with translucent overlays. The conclusion is that without underlined links the readability is improved. But still half of the participants prefer to have the used layout with blue, underlined links. The second study, which is of bigger interest here, is about the implications of links on-demand.

Two types of tasks were given to the participants. A text task where they had to answer a question concerning the content of a Web page and a link task where they had to follow a link to find the correct answer. The text task was significantly better solved with links on-demand, whereas the link task was better solved with links which were always displayed. There is a trade-off between readability of text and fast interaction with links. The user judgment is also undecided, 60% preferred links on demand and 40% rather have them always displayed.

2.5 Link augmentation with XLink

The topic of XLinks is addressed in this section. XLinks are a development of the W3C [17] and present one occurrence of link augmentation. In the following the concepts of XPath, XPointer and XLinks in general are discussed. Then we show some examples where this concepts have been used.

2.5.1 XPath

As it is a common task to address parts of an XML document, the W3C developed the XML Path Language (XPath) [14]. With this standard not every application has to define its own method for this task. XPath is now used in XSLT [28], XPointer (described in the next section) and XQuery [4]. Relating to the development of XQuery the second version of XPath is under development [3].

XPath can address any part of an XML document tree. This is done with a location path which describes step by step a specific node-set. It also provides a set of expressions and functions. They enable XPath to evaluate complex requests for a document part. Here is an example for a location path:

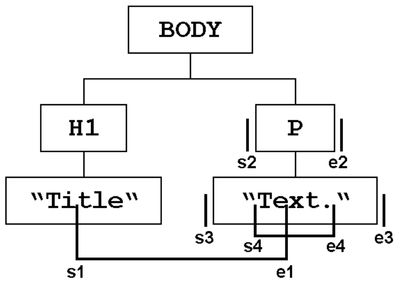
```
/descandant::book[attribute::title="XPath"]/child::chap
[position()=1]/child::sect[position()=2] ->

//book[@title="XPath"]/chap[1]/sect[2]
```

This example describes the path to the second section in the first chapter of any book with the an attribute title of value XPath. The second version is abbreviated notation to make the location path more human readable.

2.5.2 XPointer

Another interesting standard related to linking is XPointer [18, 19, 24, 25], which provides a general way to select fragments of an XML document. It was developed with regard to XLink, which is discussed in the next section.

XPointer cannot only access sets of elements, but also specific parts of a document. Therefore, the concepts of *points* and *ranges* are defined. A Point is defined by a node called the container node and a non-negative integer, called the index. It can represent the location preceding any individual character or preceding or following any node in the information set constructed by an XML document. A range is defined by two points, a startpoint and an endpoint. The range represents all of the XML structure and content between the startpoint and the endpoint. Examples are shown in Figure 2.12. XPointer extends the XPath functions set with functions dealing with ranges.
XPointers are not only very flexible, but also pretty robust. Changes in a document often do not change an XPointer describing a specific unchanged part of a document. With the use of IDs they can even be made more robust. The problem with IDs is that the author has to provide them. Therefore an XPointer cannot use IDs if the creator is not the owner of the document or they have not already been defined.



Range	startpoint		endpoint	
	Node	Offset	Node	Offset
s1 ——— e1	Text 1	2	Text2	2
s2 ——— e2	BODY	1	BODY	2
s3 ——— e3	P	0	P	1
s4 ——— e4	Text2	0	Text2	5

Figure 2.12: Points and Ranges (according to [35, p.14])

2.5.3 XLinks

The XML Linking Language (XLink) [17] defines how hyperlinks can be used in XML-based environments. XLink uses XPointer as resource identifier. The links can either be embedded or stored in an external linkbase.

XLink supports two types of links: simple and extended links. The simple links are modelled similar to HTML's link model and are supported by some browsers like the Mozilla browser family. They do not offer new options for augmented linking since they are always inline and only provide unidirectional links. The extended links are far more powerful and offer possibilities for a more elaborate link model.

```
<extendedlink xlink:type="extended">
  <loc xlink:type="locator" xlink:href="a.xml" xlink:label="x"/>
  <loc xlink:type="locator" xlink:href="b.xml" xlink:label="y"/>
  <loc xlink:type="locator" xlink:href="c.xml" xlink:label="z"/>
  <loc xlink:type="locator" xlink:href="d.xml" xlink:label="z"/>

  <go xlink:type="arc" xlink:from="x" xlink:to="y"/>
  <go xlink:type="arc" xlink:from="y" xlink:to="z"/>
</extendedlink>
```

Extended links mainly consist of *resources*, *locators* and *arcs*. The resources and locators define a starting point or a target, whereas the resources are inline and the locator can be external. The arcs provide the connection between them. They allow to define traversal rules and behavior. An example from [16, p.6] of an extended XLink is given above. The XLink standard supports n-ary links, are typed and provide a divers useful attributes. Separating the links form the content is also possible with them [20]. They could serve as augmented links for a web with XHTML pages, but they are not yet supported in any major browser. The W3C does not provide any guidelines for visualization either. Maybe in the future XLink will become a standard for hyperlinks, but nowadays this is definitely not the case due to the lack of application implementing the standard.

2.5.4 Examples

Some of the above discussed technologies have been used for various implementations of link systems and other XML applications. In this section we show some examples of these implementations.

Xspect

Xspect is an implementation of XLink [11, 12]. It handles transformation between an open hypermedia format (OHIF) and XLink. The Xspect system[5], which is based on XSLT and JavaScript provides users with an interface to browse and merge linkbases. It supports navigational hypermedia in the form of links inserted on the fly into Web pages as well as guided tours presented as SVG (Scalable Vector Graphics). Xspect has two implementations: one server-side and one running on the client. Both implementations provide the user with an interface for the creation of annotations.

XLinkProxy

The XLinkProxy is a Web application which allows sophisticated hyperlinks to be defined outside referring documents [13, 37]. It uses XLink and XPointer for link definition. The goal is to give users the chance to build dynamic multi-destination, multi-directional link databases. XLinkProxy offers a user interface to create and visualize links for both HTML and XML documents accessed through an HTTP proxy, whereby the HTML has to be converted into an XML Syntax. The XPointer library used for this task is designed to run on a Web server and is written in JavaScript.

XSLT Extension

The XSLT++ engine is an enhanced version of XSLT [37]. Whereas XSLT uses XPath, this solution supports XPointer instead of XPath for pattern selection. With XSLT++ it is possible to apply transformation to portions of text and not only to elements, attributes or text nodes. The current implementation of XSLT++ is written in Java and is based on Xalan[6], the XSLT engine by the Apache group.

XConnector

Another example is XConnector [31]. This is a language for the creation of complex hypermedia relations with causal or constraint semantics. XConnector, like iServer, allows the definition of relations independently of which resources are related. Another feature is the specification of relation libraries, providing reuse in relationship definition. The main goal is to improve linking languages or the linking modules of hypermedia authoring languages in order to provide multimedia synchronization capabilities using links.

[5]http://fahbentor.daimi.au.dk
[6]http://xml.apache.org/xalan-j/

3 iServer in the context of link augmentation

In this chapter we discuss how iServer can be used for link augmentation. Therefore, the iServer model and also three possible approaches for the implementation of an XHTML plug-in are discussed.

The iServer is an integration server architecture developed by the Global Information Systems group[1] at ETH Zurich. The iServer enables cross-media linking based on an object-oriented hypermedia model. This mixed-media platform can easily be extended to support new types of digital or physical multimedia resources. To use a new resource type within the platform, a plug-in has to be provided. With the use of an XHTML plug-in, iServer presents a very flexible and extensible tool for link augmentation which also offers the possibility to extend the Web to a real open hypermedia system.

3.1 The iServer Model

This section presents the main features and concepts of the iServer model [36]. These are links, resources and selectors, layers, user management and finally the plug-in mechanism.

The model is based on the semantic, object-oriented data model OM. The shaded rectangular shapes denote collections of objects where the name of the collection is given in the unshaded part. The shaded oval shapes represent associations between entities of two collections which can be restricted by cardinality constraints.

[1] http://www.globis.ethz.ch/research/iserver/

3.1.1 Links

Links within the iServer architecture are always directed and are bound to one or more sources und lead to one or more targets (Figure 3.1). An information entity can be used equally as a link source or a link target. This is taken into account in the information model by introducing a generic entity concept.

The cardinality constraints `1:*` specified at the source and target points of the associations indicate that each link must have at least one and possibly many sources and targets. With this the support of multi-headed links and multiple sources is achieved. The `0:*` at the target point of both associations specifies that there is no limit on the number of links for which an entity may be the source or target.

The representation of `Links` as a subcollection of `Entities` also allow to define links between links. Additionally, each entity can be associated with a set of properties as specified by the `Properties` collection. These properties are represented by key and value pairs. While they are not predefined by the iServer framework, they can be defined individually. For example to customize the behavior for specific application domains similar to the behavior attributes of XLink.

An other important fact is that the underlying OM model provides bidirectional associations as a higher level construct, all the associations used within the iServer framework are also bidirectional. This allows to get corresponding link sources for a given target object.

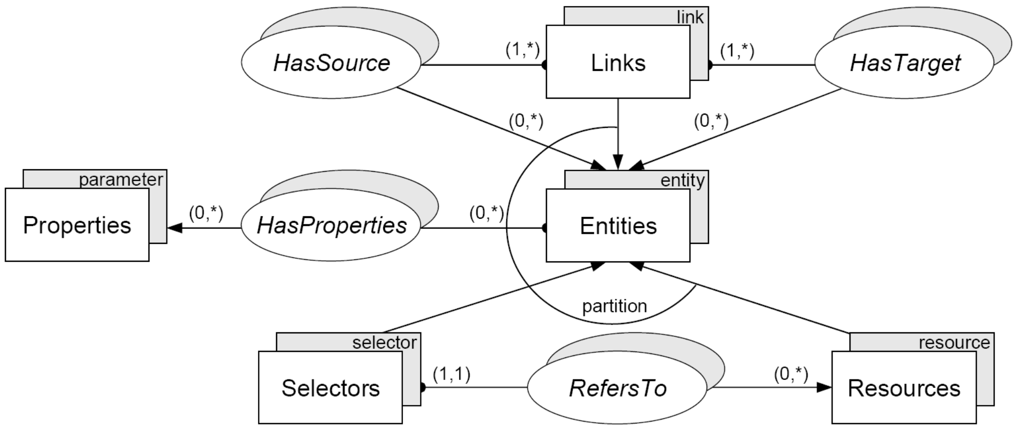


Figure 3.1: iServer links

3.1.2 Resources and Selectors

The simplest type of entity is the resource type, which represents an entire information unit. Since authors want a better control of the link granularity, there is also the possibility to address specific parts of a document. A `selector` is an other subtype of entity which allows to address parts of the related `resource`.

Each media type can define specific media resource and selector subtypes. `Resources` can be any kind of multimedia documents like videos, pictures, sounds, web documents, but are not limited to these. Physical objects like paper documents are supported too. The selector depends on the `resource` type. It can address a paragraph in a text document, a shape in a picture, a time-span in a video or something else.

3.1.3 Layers

The iServer framework uses `layers` to provide the possibility of overlapping selectors and become more flexible in defining the semantics of link anchors. Each `selector` is associated with exactly one layer and no overlapping within a layer is allowed (Figure 3.2). `Layers` are explicitly ordered and therefore priorities can be assigned. In the case of overlapping `selectors` the uppermost `selector` will be selected. The layers may be activated and deactivated which allows to generate a content-specific set of links.

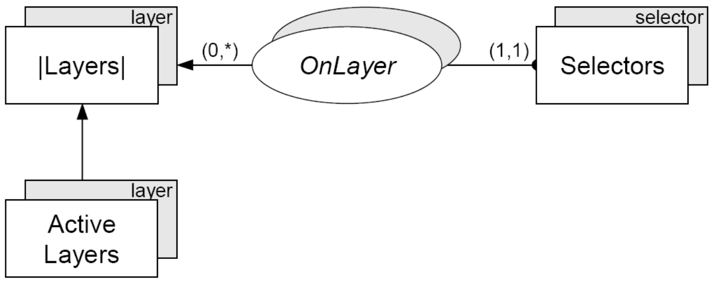


Figure 3.2: iServer layers

3.1.4 User Management

The iServer framework provides user management as one of the fundamental components (Figure 3.3). A `user` can either be an `individual` or a `group`. Each entity is created by exactly one individual `owner`. He has full control over an entity's content and can define access rights for other `groups` of `users` or `individuals`. With the help of the two associations `AccessibleTo` and `InaccessibleTo` the rights can be defined in a flexible way. In addition, `users` can store any number of preference `parameters`.

3.1.5 Plug-in mechanism

A new media type can only be used within iServer once a corresponding plug-in is implemented. This extension consists of types for both the media `resource` and the media `selector` and some functionality for manipulating them. Preferably, an access API is provided to simplify the interaction with the new type.

The plug-in has to provide two classes to extend the iServer framework to support the new media type:

- a specified resource class extending the iServers `Resource` class describing object properties and operations

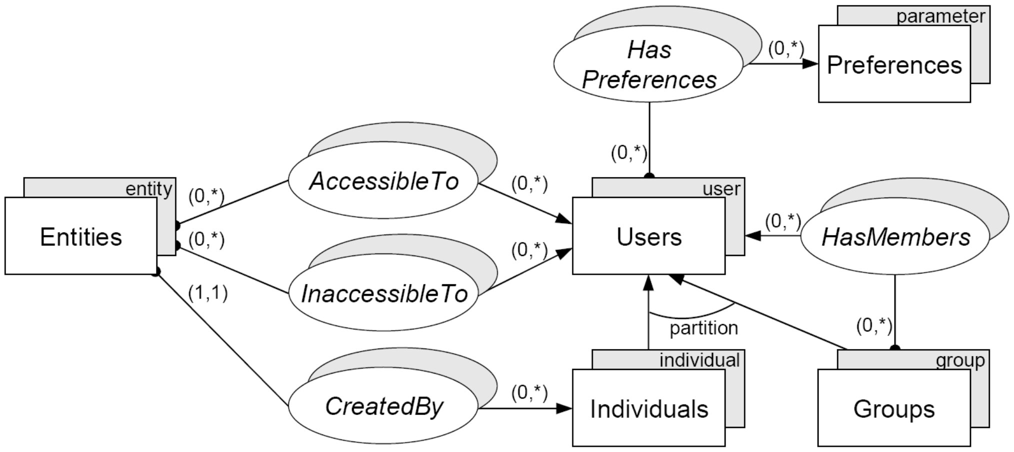


Figure 3.3: iServer user management

- a specified selector class extending the iServers `Selector` class and specifying how to address specific parts of the new media type

3.2 Architectures for the XHTML plug-in

After presenting the iServer model this section focuses on the possible approaches for the XHTML plug-in. Three different approaches can be distinguished: The proxy server approach, the use of browser plug-ins and the authoring tool integration. Despite the major differences between them, a combination of different approaches remains an option which could remove some of the limitations. We discuss the features and possible tools as well as advantages and disadvantages of each approach.

3.2.1 Proxy server approach

In this approach a proxy server is used to interact with the existing Web architecture. Instead of working directly with a Web Client or a Web Server, the proxy server is installed between the Web client and the WWW. For example Jigsaw[2] a Java based proxy server is an option.

The proxy is connected with iServer to retrieve linking information and if needed add new links. It can filter HTTP requests and modify the responses of the Web servers. Additional code and for example JavaScript functions can be implemented in a Web page. This can be used for visualizing iServer links and offering authoring functionality in the browser. For authoring also the use of signed Java applets which communicate directly with the iServer provide a feasible solution. This approach was chosen in a previous project to build an XHTML plug-in for iServer described in [35].

Advantages of this approach are as follows:

- Web client independence (proxy support is required)
- platform independence (with a Java based proxy)
- already existing work on an XHTML plug-in

And the disadvantages are:

- limited functionality of link visualization and authoring in a Web browser
- no direct support for linking with other media types
- delay of page display due to the modification in the proxy

3.2.2 Browser plug-in

A browser plug-in could enable a standard Web browser to be able to display and manipulate iServer links. The plug-in can communicate directly with a local or remote iServer instance. The visualization of the links can be done by the browser itself allowing to do it after a page has loaded.

The plug-in can be written for Mozilla Firefox, Microsoft Internet Explorer or any other browser allow plug-in support. Both browser Mozilla Firefox and Microsoft Internet

[2]http://www.w3.org/Jigsaw/

Explorer offer sufficient support for plug-ins.

Firefox has the advantages of being open source, cross-platform, offering many examples of existing plug-ins and extensions, support for XPCOM and a big active developer community. Shortcomings are the still limited market share and fast changing versions.

Internet Explorer has its strengths in the huge market share, a very good support of COM and Active-X components. Disadvantages are that it is relatively old (October 2001), that a completely new release is planned (end 2005) and the platform dependency.

Advantages of the browser plug-in approach are as follows:

- a standard Web client can be used for browsing
- many forms of link visualization can be exploited
- seamless integration in existing Web architecture

And the disadvantages are:

- plug-in is restricted to one specific Web client
- no direct support for linking with other media types, an integration with the authoring tool remains a possibility

3.2.3 Authoring tool integration

An authoring tool for the iServer framework, allowing to create links between different media types, is under development. Like iServer, it allows plug-ins for different media types. Such a plug-in for visualization and authoring of iServer links in XHTML is possibility for the implementation.

For the integration a Java browser is needed to display the content of the XHTML Web pages. Proprietary Java browsers like ICEbrowser[3] or Clue Web Browser[4] do not seem to be a satisfying solution. Possible open Java browsers which could be integrated are:

- Jazilla[5] is a clone of the Mozilla browser completely written in Java
- HotJava[6], the sun browser is unfortunately not further developed and supported any more
- JRex[7] is a Java browser component with a set of API's for embedding Mozilla browser within a Java application

Advantages of this approach are as follows:

[3]http://www.icesoft.com
[4]http://www.netcluesoft.com
[5]http://jazilla.mcbridematt.dhs.org
[6]http://java.sun.com/products/archive/hotjava/index.html
[7]http://jrex.mozdev.org

- direct support for links with other media types
- layer and user management already implemented and could be used
- easy interaction with iServer

And the disadvantages are:

- a non-standard browser has to be used within the authoring tool
- browsing of iServer links is only possible with the authoring tool

4 Architecture

In this chapter we describe the overall architecture choosen for the iServer XHTML plug-in. Therefore an overview of the used components is given. The iServext component for link visualization and authoring is then described in more detail.

From the possible architectures we choose the browser plug-in approach as described in Section 3.2.2. It offers the most possibilities in the area of link visualization and it can be integrated in existing Web technologies. The browser of choice was Mozilla Firefox mainly because of the open architecture und the easy integration of extensions.

4.1 Overview

This section gives an overview of the architecture. The system is composed of the following parts:

- iServext, the iServer extension for Mozilla Firefox
- iServer, enhanced by the XHTML plug-in, used by the iServer Web Service to access the link database
- iServer Web Service, used by the iServext to access the link database and create new links

The iServext component is discussed in the next section. It communicates with the iServer Web Service and the authoring tool. The iServer Web Service component provides the possibility to access a link database through a Web Service. Its implementation and architecture is described in [15]. The iServer component was described earlier in Section 3.1. A visualization of the relations is shown in Figure 4.1.

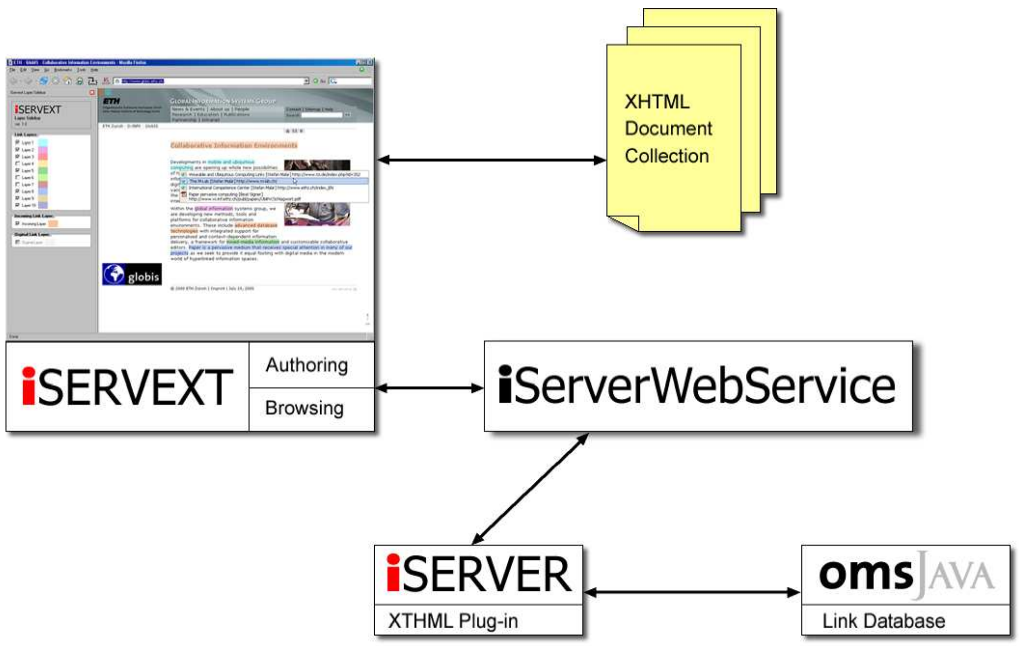


Figure 4.1: Architecture overview

4.2 The iServext component

In this section we describe the iServext component. It has two main tasks:

- The integration and visualization of link information from the iServer Web Service into a corresponding document
- The authoring of new links between XHTML documents respectively XHTML Selectors

Both tasks are independent, but communicate with the same external components. They are also integrated into one component, which helps to reduce the number of different components.

4.2.1 Link integration and visualization

The different components needed for the link integration and the flow of information between them are shown in Figure 4.2.

The iServer links have to be inserted into a corresponding document when it is loaded in the browser (①,②). Therefore, the iServext sends a SOAP request ③ to the iServer Web Service requesting all existing links for the concerning current document.

The iServer Web Service either can be local or remote. After receiving the SOAP request it fetches the link information with the help of the iServer library from the link database (④,⑤), which provides all the information (⑥,⑦) required for the SOAP response to the iServext.

The SOAP response ⑧ contains all the necessary information about the `Links` and `Selectors`. Thus the iServext can add the links in the form of layers directly into the DOM Tree of the document ⑨. They are displayed by the rendering engine without reloading the document.

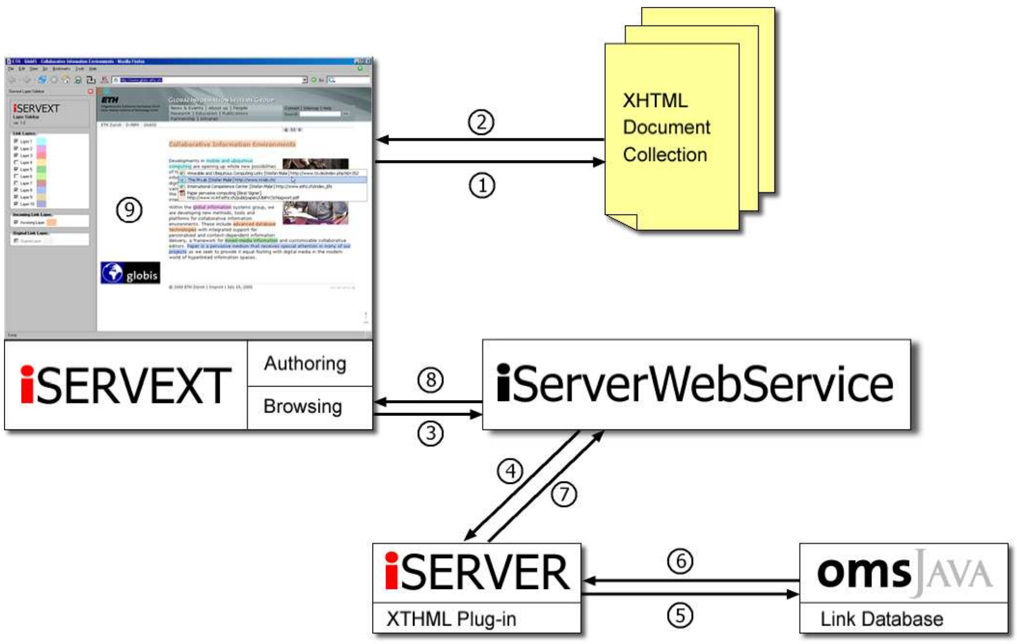


Figure 4.2: Architecture of link integration and visualization

4.2.2 Link authoring

The components used for the authoring of links are shown in Figure 4.3. The flow of information is indicated as well.

For the authoring of XHTML iServer links a `Ressource` and a `Selector` is needed. The iServext provides the functionality for retrieving the URL as `Ressource`-identifier and an XPointer as `Selector` from any given selection. In the authoring sidebar one can define a link with a source and a target①. You can name the `Ressource` and the `Selector` as well as the `Link`.

The link information is submitted to the iServer Web Service through a SOAP request ②. The iServer Web Service creates a new link in the link database with help of the iServer library (③,④). If the creation of the link was successful this information is sent to the iServer Web Service(⑤,⑥) and converted into a SOAP response which is sent to the iServext ⑦.

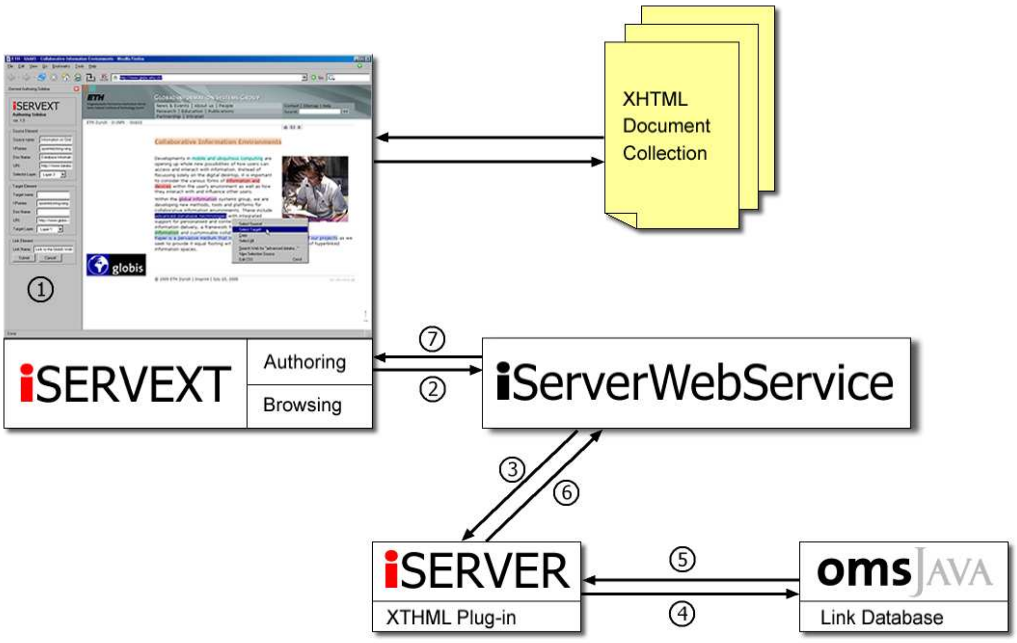


Figure 4.3: Architecture of link authoring

5 Implementation

In this chapter we present various aspects of the iServext implementation. In the following sections the GUI integration of the iServext, the XPointerLib, SOAP[1] which was used for the communication with iServer Web Service, and the details of the integration of existing links are described. We also discuss the implementation of the authoring functionality with the authoring sidebar and the corresponding iServer Web Service methods used for it in the last section of this chapter.

As a Mozilla Firefox extension, the iServext is based on JavaScript and XUL[2]. The use of XPCOM[3] Services is possible too, as it is done here with the XPointerLib. In the next section we discuss how the integration into the GUI of Firefox was realized.

5.1 iServext GUI integration

How to integrate the iServext into the Firefox GUI is discussed in this section. First we describe the integration of the control items into the overlay of the Firefox application. In the second subsection the implementation of the preferences window is discussed.

5.1.1 Overlay integration

For the overlay integration basically a XUL file called `overlay.xul` is loaded into the chrome environment to make the control elements visible in the Firefox. Chrome is the user interface part of the application window that are outside of a window's content area. Toolbars, menu bars, progress bars, and window title bars are all examples of elements that are typically part of the chrome.

[1]Simple Object Access Protocol, further information: http://www.w3.org/TR/soap/
[2]XML User Interface Language, further information: http://www.xulplanet.com
[3]Cross Platform Component Object Model, further information: http://www.mozilla.org/catalog/architecture/xpcom/

For the iServext menu items are inserted into the context menu, the sidebar menu and the tools menu. The integration of the saveDB method into the tools menu is shown below as an example. The outer tag defines which menu should be used and the inner tag what should be inserted.

```
<menupopup id="menu_ToolsPopup">
  <menuitem  id="iservext_toolmenu"
  insertbefore="devToolsSeparator" label="&iservext; &savedb;"
  accesskey="&savedbkey;" oncommand="saveDB();" />
</menupopup>
```

Another possibility of integration is used for the method to switch the iServext on and off. An icon can be added to the toolbar palette. The code used for this is shown here:

```
<toolbarpalette id="BrowserToolbarPalette">
  <toolbarbutton id="iservext_toolbar" class="toolbarbutton-1"
  label="&iservext;" oncommand="onOff();"
  tooltiptext="&onofftooltip;"/>
</toolbarpalette>
```

The picture of the icon is defined in a CSS file called `iservext.css` which is stored in `chrome://iservext/skin/`. All the methods used in the `overlay.xul` are defined in the JavaScript file `overlay.js` which is included into the `overlay.xul` together with all the other JavaScript files used in the extension. The entities used in the XUL are defined in the `iservext.dtd` which has to be included as well and the properties for the strings in the JavaScript files are in `iservext.properties` (see also Appendix A.2 for file locations).

5.1.2 Preferences window

The preference window is used to set the various options of the iServext. The URL of the preference window can be defined in the `install.rdf` by adding the following line:

```
<em:optionsURL>
   chrome://iservext/content/prefs.xul
</em:optionsURL>
```

In the `prefs.xul` all the information about the visualization and the handling of the preferences is contained. With the use of the `<prefwindow>` tag the XUL allows the manipulation of preferences with a very simple syntax.

A preference value is called by a tag `<preference>`. An example for the integer opacity is shown here:

```
<preference id="opacity" name="extensions.iservext.opacity"
type="int"/>
```

The preference then can be displayed in a simple textbox and if the value is changed and submitted the preference is changed. The sample code for the preference opacity from above is the following:

```
<label value="&background;"/>
<textbox id="text_opacity" preference="opacity" width="30"/>
<label value="%"/>
```

The preference attribute contains the id of the `<preference>` tag. Boolean values can be displayed as radio buttons and for colors a ¡colorpicker¿ tag is available.

The default values of the preferences are defined in the JavaScript file `preferences.js` by using the syntax shown below. Three types of preference values are distinguished: integer, boolean and strings.

```
pref("extensions.iservext.opacity", 30);
pref("extensions.iservext.createlinklayer", false);
pref("extensions.iservext.bgcolorlink", "#999999");
```

5.2 XPointerLib

As an other implementation issue we discuss the XPointerLib[4] in this section. The XPointerLib is an XPCOM Service which implements XPointer for the Mozilla browser family. It was originally motivated by the Annozilla Project[5] and then was developed as a standalone project. Since 2003 there are no new releases. Most likely because Mozilla provides native support for some XPointer functionality[6]. For our purpose the native support is not sophisticated enough yet because points and ranges are not supported.

The XPointerLib Service can be called in JavaScript as follows:

```
var xptrService = Components.
   classes["@mozilla.org/xpointer-service;1"].getService();
xptrService = xptrService.
   QueryInterface(Components.interfaces.nsIXPointerService);
```

The `xptrService` implements several methods from which the following are used in the iServext:

- `createXPointerFromSelection` creates an Xpointer from a given selection.
- `parseXPointerToRange` parses an XPointer for a given document into a Mozilla range object.
- `markElement` marks an element to be ignored by the XPointer parser. In this way they do not interfere with other XPointers of this document.

Examples of the the JavaScript code for these methods are given below:

```
var xptrString = xptrService.createXPointerFromSelection(
                  selector(),focus_selector().document);

var range = xptrService.parseXPointerToRange(xptrString,
            focus_selector().document);

xptrService.markElement(insertedNode);
```

The `selector()` method returns the current selection and the `focus_selector()` returns the currently focused window. The `createXPointerFromSelection` method is used for the creation of link `Selectors` and the `parseXPointerToRange` method is needed for the integration of the links into a corresponding document. For all the nodes inserted into DOM tree the `markElement` method is used.

[4]http://xpointerlib.mozdev.org/
[5]http://annozilla.mozdev.org/
[6]http://www.mozilla.org/newlayout/xml/

The XPointer which is created can be in the forms presented below. It consists of just a string-range or a start-point and a end-point. The second case is used if the XPointer includes more than one node. Both the start and end-point contain a string-range with the following arguments: An XPath expression, an empty argument, the text start index and the text offset.

```
xpointer(string-range(/html[1]/body[1]/p[1], "", 3, 5))

xpointer(start-point(
string-range(/html[1]/body[1]/p[1], "", 6,1))
/range-to(end-point(
string-range(/html[1]/body[1]/p[1]/strong[1],"", 3, 1))))
```

5.3 SOAP - communication with the iServer Web Service

In this section the communication with the iServer Web Service is discussed. First we describe the methods offered by the Web Service and then the methods used by the iServext for communication with the Web Service.

In Figure 5.1 we give an overview of the methods called when a new document is loaded. The main methods in this overview are described in this and the next section of this chapter.

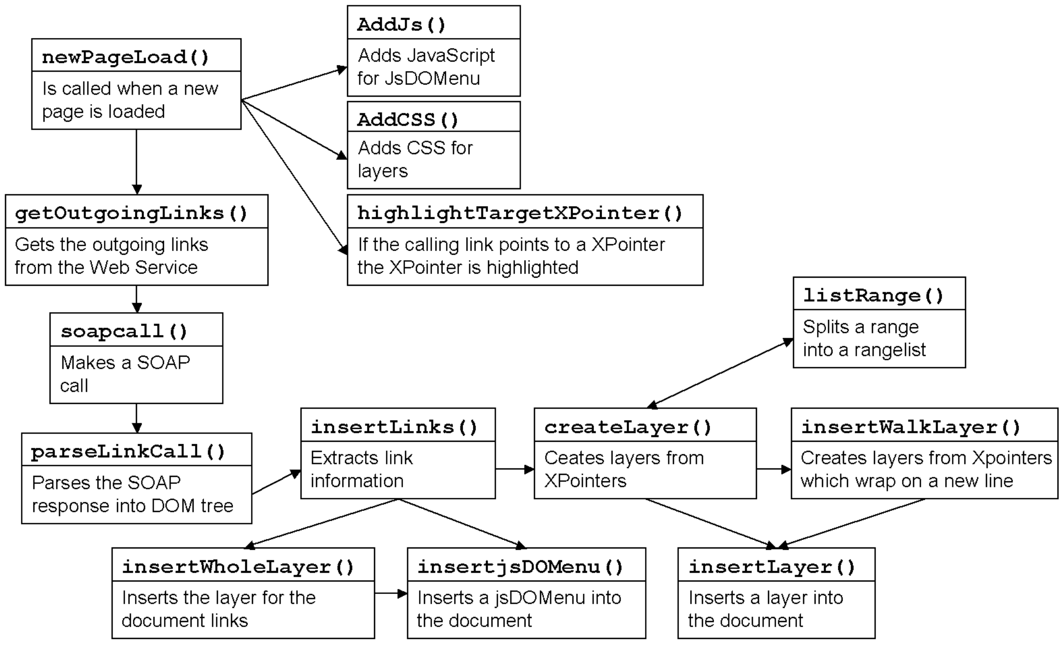

newPageLoad()
Is called when a new page is loaded
AddJs()
Adds JavaScript for JsDOMenu
AddCSS()
Adds CSS for layers
highlightTargetXPointer()
If the calling link points to a XPointer the XPointer is highlighted
getOutgoingLinks()
Gets the outgoing links from the Web Service
soapcall()
Makes a SOAP call
parseLinkCall()
Parses the SOAP response into DOM tree
listRange()
Splits a range into a rangelist
insertLinks()
Extracts link information
createLayer()
Ceates layers from XPointers
insertWalkLayer()
Creates layers from Xpointers which wrap on a new line
insertWholeLayer()
Inserts the layer for the document links
insertjsDOMenu()
Inserts a jsDOMenu into the document
insertLayer()
Inserts a layer into the document

Figure 5.1: Overview of methods called on pageload

5.3.1 iServer Web Service methods

The iServer Web Service offers a set of basic iServer methods implementations and was extended with new methods to serve the XTHML Plug-in. The iServext uses the following four methods:

- The `getOutgoingLinks(String documentUri)` method returns all outgoing links for a give XHTML document. The links can either link directly from the document or from a selector that has been defined for the document. It is called when a new page is loaded.
- The `getIncomingLinks(String documentUri)` method returns all incoming links for the XHTML document. The links can either link directly to the document or to a selector that has been defined for the document. It is called when a new page is loaded and the preferences are set to display the incoming links.

- The `createXHTMLLink(Strings: linkName, sourceDocumentName, sourceDocumentUri, sourceSelectorName, sourceSelectorXpointer, sourceSelectorLayer, targetDocumentName, targetDocumentUri, targetSelectorName, targetSelectorXpointer, targetSelectorLayer, creatorName)` method creates a new XHTML link based on source and target information. It is called by the iServext Authoring Sidebar.
- The `saveDB()` method commits the current state of the link database.

All the methods of the iServer Web Service are called through a SOAP request and give back a SOAP response. How the iServext handles SOAP is described in the next subsection.

5.3.2 iServext methods for communication

As mentioned earlier SOAP requests are used by the iServext for the communication with iServer Web Service. Firefox offers built-in JavaScript methods for SOAP communications which are described in this subsection together with the methods the iServext uses to get the link information.

When a new page is loaded a listener calls the newPageLoad() method. This method calls methods to add the JavaScript for the jsDOMenu and the CSS for the layers. Furthermore, it executes the `getOutgoingLinks()` method and if the preference are set for it the `addLinkLayer()` and the `addIncomingLayer()` method.

The `addLinkLayer()` methods adds a layer over the existing links that they can still be accessed when a new link lays on the of them. The `getOutgoingLinks()` and the `addIncomingLayer()` method work alike and the only difference is that the first one gets the outgoing links and the second one the incoming links. So we will only describe how the `getOutgoingLinks()` method works.

The `getOutgoingLinks()` method gets the URI of the loaded document and sets it as parameter for a SOAP call. The code used for this is shown here:

```
var params = new Array();
params[0] = new SOAPParameter(documentUri,"documentUri");

var inText = 'getOutgoingLinks';
soapcall(inText,params,parseLinkCall,'outgoing',aDoc);
```

The `soapcall()` method shown below creates a SOAP. It uses the built-in method `SOAPCall()` and adds the appropriate attributes to it. Then the call is invoked by a function which calls a SOAP response handler method and the callback method which is in this case the `parseLinkCall()` method.

```
function soapcall(aMethod,aParams,aCallback,aTyp,aDoc){
  var soapCall = new SOAPCall();
  var serverURI = getPrefBranch().getCharPref('servername');
  soapCall.transportURI = serverURI;
  var serviceName = getPrefBranch().getCharPref('servicename');
  soapCall.encode(0, aMethod, serviceName, 0, null,
                 aParams.length, aParams);
  var currentRequest = soapCall.asyncInvoke(
     function (response, soapcall, error)
     {
        var r = handleSOAPResponse(response,soapcall,error);
        if(aDoc){
            aCallback(r,aTyp,aDoc);
        }
        else{
            aCallback(r,aTyp);
        }
      }
  );
}
```

The `parseLinkCall()` method parses the result of the SOAP response into a DOM tree and calls the `insertLink()` method or returns an error if something went wrong. The main code for the extraction of the XML from the response and the parsing into a DOM tree is shown here:

```
 var params = aResult.getParameters(false,num);
 var xml = params[0].value;

 var parser = new DOMParser();
 var dom = parser.parseFromString(xml, "text/xml");
```

How the `insertLinks()` method works is described in the next section. The methods used for the communication with SOAP are stored in the `soap.js` JavaScript file.

5.4 Integration of links

The integration of the links into the XHTML documents and the methods used for it are discussed in this section. The handling of the XPointers and the visualization of the links are presented in the next two subsections. We also present the layer sidebar of the iServext in the last subsection.

5.4.1 Handling of the XPointers

From the iServer Web Service all the links and information about them are parsed into a DOM tree as described in the Section 5.3. The `insertLinks()` method loops over all the links in the DOM tree extracts the information about them. If the source of a link is an `XHTMLDocument` the link information is written to an array. After the looping is finished the `insertWholeLayer()` method inserts one layer on the top of the document with all the links from this array. Otherwise, if the source is an `XHTMLSelector`, a layer is inserted by the `createLayer()` method over the XPointer area.

The XPointers are parsed to ranges with the `parseXPointerToRange` method described in Section 5.2. Each range then is split into a list of ranges which have the startContainter equal to the endContainer. When this is done they can be surrounded by a tag. The `listRange()` method which does this is based on the Annozilla project, but was enhanced with new functionality. The handling of `<img>` and `<br>` tags is now supported as well as the handling of tables. The `listRange()` method uses the `evaluateNode()` method to check if a node is completely covered by a range. If it is not a text node the children are visited and checked themselves. Otherwise, if a text node is just partly covered, it is split into a covered node and not covered node for the further handling. These and some additional methods are stored in the `ranges.js` JavaScript file.

The `createLayer()` method which calls the `listRange()` described above uses the list of ranges to create layers over each range in the list and so cover the selection of the XPointer it was composed from. Basically this is done by putting a `<span>` around the elements and computing the position as well as the size of them. If an element wraps onto new line, all the parts of it are evaluated by the `insertWalkLayer()` method and for every line a separate layer is created. The `insertLayer()` method, partly shown below, then creates a div layer and sets the appropriate attributes for it. These are the position and size stored in the `aOrgElePos` class and the behavior stored in the `aLayerAttrs` class. Afterwards the div layer is inserted into the DOM tree of the document and marked that it is ignored by the XPointer service.

```
function insertLayer(aLayerAttrs,aOrgElePos,aDoc){

    //calls xpointer servcie
    var xptrService = Components.classes["@mozilla.org/
                      xpointer-service;1"].getService();
    xptrService = xptrService.QueryInterface(Components.
                  interfaces.nsIXPointerService);
    //create the div tag
    var div = aDoc.createElementNS(
              "http://www.w3.org/1999/xhtml","div");
    //set all attributes
    div.setAttribute("id",aLayerAttrs.id);
    div.setAttribute("class",aLayerAttrs.class);
    div.setAttribute("style","left:"+aOrgElePos.xpos+"px;
                    top:"+aOrgElePos.ypos+"px;
                    width:"+aOrgElePos.elewidth+"px;
                    height:"+aOrgElePos.eleheight+"px;");
    if(aLayerAttrs.onClick){
    div.setAttribute("onclick",aLayerAttrs.onClick);
    }
    ...
    if(aLayerAttrs.title){
        div.setAttribute("title",aLayerAttrs.title);
    }
    //insert the tag into the body
    var bodyNode = aDoc.getElementsByTagName("body")[0];
    bodyNode.appendChild(div);
    xptrService.markElement(div);
}
```

This and some additional methods are stored in the `iservext.js` JavaScript file.

5.4.2 Visualization

The visualization of the layers is done through CSS[7] and absolute positioned div layers. This allows a very flexible handling of the link layers and the multiple layers on top of each other.

The CSS is added by the `addCSS()` method. It gets the information about the link color preferences and adds stylesheet information about the layers and for the various mouse pointers which are used to visualize the different types of layers, the direct links and the indirect links. The indirect links use the the jsDOMenu to show the different targets. The CSS for this JavaScript pop-up menu is added by the `addJsDomenu(aDoc)` method. All this CSS code is added directly into the DOM tree of the document when it is loaded. If neccessary layer styles for the incoming link layer and the original link layer are added as well. The methods used for adding the CSS are stored in the `css.js` JavaScript file.

The div layers are simple `<DIV>` tags added to the DOM tree. The include all the information about the link and are positioned absolute over the XPointer area they belong to. They are also on different z-index levels according to the iServer layer they belong to. Since they are transparent, the text and other layers below it are still visible. But only the topmost can be accessed by clicking on it. To access the layers below the layer above has to be hidden. This can be done by the Layer Sidebar which is described in the next subsection.

[7]Cascading Style Sheets, more information: http://www.w3.org/Style/CSS/

5.4.3 Layer Sidebar

The Layer Sidebar is used to display all names and colors of the iServer layers. It allows also to deselect a layer and thus hide it. The user interface of a Firefox sidebar is written in XUL and the functionality is done by JavaScript methods.

The Layer Sidebar registers a listener which is shown below when it is loaded. It reloads the sidebar whenever a new document is loaded or the location is changed which means in this case the tab is switched. This is done that always the correct layers for a document are shown as selected.

```
var sidebarListener = {
 QueryInterface: function(aIID)
 {
  if(aIID.equals(Components.interfaces.nsIWebProgressListener)
  ||aIID.equals(Components.interfaces.nsISupportsWeakReference)
  ||aIID.equals(Components.interfaces.nsISupports))
  return this;
  throw Components.results.NS_NOINTERFACE;
 },
 onStateChange: function(aProgress, aRequest, aFlag, aStatus)
 {
  if(aFlag & STATE_STOP){
     // This fires when the load finishes
     top.document.getElementById('sidebar').reload();
  }
  return 0;
  },
  onLocationChange: function(aProgress, aRequest, aURI)
  {
    // This fires when the location bar changes i.e load event
    // is confirmed finished or when the user switches tabs
    top.document.getElementById('sidebar').reload();
  },
}
```

The settings for the layers in the sidebar is done dynamically by the `getLayerPrefs()` and the `setLayerPref()` method. They get the colors of the layers form the preferences and the visibility of them from the document. The attributes of the the tags in the XUL file can be accessed and altered through the DOM tree of it.

By clicking on a layer in the Layer Sidebar it can be selected and deselected the state is shown in a checkbox in front of the layer name. If a layer is deselected it is hidden in the browser window by the `changeLayer()` method which alters the visibility style attribute of it. This may help to increase the clarity if a document has many links.

5.5 Authoring

In this section the authoring functionality of the iServext is presented. First the implementation of selection methods and then the main tool, the Authoring Sidebar, are discussed.

The selection of a source or a target is done through the right click context menu of the browser. There we offer the new menu items `Select source!` and `Select target!`. They call the `targetXPointer()` respectively the `sourceXPointer()` method which check if the Authoring Sidebar is open and if it is not open a prompt window is shown and asks the user if he wants to open it. Afterwards they call the insert `insertSourceValues()` method which is shown below. This method creates an XPointer from the selection with help of the XPointerLib (see Section 5.2). Aftwards it copies the XPointer and the URI of the XHTML document into the Authoring Sidebar. If no selection is made only the URI is copied.

```
function insertSourceValues(aType){

  // Calls xptrService
  var xptrService = Components.classes
    ["@mozilla.org/xpointer-service;1"].getService();
  xptrService = xptrService.QueryInterface(
              Components.interfaces.nsIXPointerService);
  var xptrString = xptrService.createXPointerFromSelection(
                 selector(), focus_selector().document);
  var urlString = url_selector();

  //if nothing is selected empty xptrString
  if(xptrString == 'xpointer(/html[1])'){
    xptrString = '';
  }

  if(aType == 'source'){
    top.document.getElementById('sidebar').contentDocument.
     getElementById('sourceSelectorXpointer').value=xptrString;
    top.document.getElementById('sidebar').contentDocument.
     getElementById('sourceDocumentUri').value=urlString;
  }
  else if(aType == 'target'){
   ...
  }
}
```

The naming and allocation of layers is then done in the Authoring Sidebar. The implementation of it is described in the next subsection.

5.5.1 Authoring Sidebar

The Authoring Sidebar is used to create new links between XHTML documents. Like the Layer Sidebar described above the user interface is written in XUL and the functionality is done by JavaScript methods.

The interface of the Authoring Sidebar is like a form, but since XUL does not offer form functionality like for example HTML, the submission is done by JavaScript methods. The input fields are simple XUL textboxes like the one shown here:

```
<textbox id="sourceSelectorXpointer" width="100"/>
```

When the form is submitted the `checkForm()` method checks if all the required fields are filled and then the `createXHTMLLink()` method accesses the textbox values through the DOM tree as shown here:

```
var sourceSelectorXpointer =
    document.getElementById('sourceSelectorXpointer').value;
```

Afterwards all the values are written into a parameter array which is submitted through a SOAP request to the `createXHTMLLink()` method (see Section 5.3) of the iServer Web Service. If the link is successfully created it is sent back in a SOAP response and a pop up window gives a confirmation to the user.

6 Conclusions

In this chapter we draw conclusions from what we have done throughout this semester work. Furthermore, we discuss the limitations of the current implementation and possible areas for future work.

In our work we presented some insight into the topic of link augmentation in the theoretical part. We have discussed current research projects and ideas for link visualization. We have also looked on the possibilities offered by XLinks, although the current state of them did not suit all our needs for our project, they open up interesting new prospects for the future. Furthermore, the iServer framework was described and put in the context of link augmentation.

The main goal of the semester work, as described in Chapter 1, was to develop a XHTML plug-in for iServer. This was done successfully with an architecture using a Firefox extension for the integration and visualization of XHTML iServer links as well as for the authoring of them. We have described the details about the architecture and the implementation in the Chapters 4 and 5.

During this work we were able to get to know the possibilities offered by Firefox extension and the related technologies based on the Mozilla framework. The framework is suitable for rapid implementation of Web augmentation projects which can be seamless integrated into the Mozilla product family.

We were also able to show that without to much effort link augmentation on the Web could be done. Techniques like multi destination links, additional information about link targets and others could be implemented with little enhancement of the currently used components.

6.1 Limitations

Although the main goals of this semester work have been reached some limitations of the implementation remain. We will point them out in the following list. Some of them should be removed with the enhancement discussed in the next section about future work.

- Only XHTML links are supported. Links to or from `Resources` or `Selectors` other then XHTML Documents and XPointers cannot be displayed by the iServext and no such links can be created.
- Frames are no supported. Websites which work with frames do not display the links correctly and therefore the authoring of links to or from such websites is not allowed.
- Limitations of the XPointerLib. The XPointers of the XPointerLib can not resolve all the selection made in a browser. Selections like the one with a picture in the beginning and then some other elements cause problems.
- Unresolved issue with overlapping of selectors on a layer. If selectors overlap on a layer only one of them can be accessed correctly in area in which they are intersecting.
- The incoming links are displayed after a time delay. Because the iServer Web Service crashes if two simultaneous calls are made the call for the incoming has to be made after a delay of three seconds and therefore delays the visualization of them in a document.

6.2 Future Work

The idea to build an iServer framework with plug-ins to link all kinds of media types together and navigate between is not fully reached by the XHTML plug-in architecture presented in this work. To achieve this some of the following suggested ideas for future work would have to be investigated. We also discuss some other ideas for improving the iServext.

An important issue is support of other link types then XHTML links. One point is the visualization of them and the navigation to an other media type. This would need an enhancement of the integration and visualization mechanism of the iServext. A second point is the authoring of such links. Since this authoring is not intended to be done within the iServext, an external authoring tool component should be used to handle the link creation. Such a tool is currently under development. It directly accesses and manipulates the link database. Some kind of interaction with the iServext will be necessary for the source and target selection, but because the development is not finished yet, this has to be defined later.

For the development towards the collaborative information sharing like the iServer P2P [15] some kind of user authentication is needed. This is not only an issue of future work for the XHTML Plug-in, but also for the whole iServer framework.

There are also more specific areas within the iServext where future work could be done. An idea is to provide dynamic layers instead of the fixed ten layers. Beforehand the layering concept should be thought over also in the prospect of the work with the iServer P2P architecture. Another direction of future could be that layers below other layers are made accessible directly for example through a double click. In a later step the iServext could also be localized for other languages then English. This is not really a technical issue but rather a matter of translation.

Furthermore, the current limitations could be removed if they not already will be by the future work described above. In the case of the XPointerLib the removal of the limitations might be done by the Mozilla community, but at the moment no development on the XPointerLib is done. It is intended that in a future version of the Firefox XPointers are supported natively and this functionality could then be used instead.

6.3 Acknowledgements

First of all, I would like to thank Dr. Beat Signer for supervising my semester work. He gave me the freedom to choose my own ways to reach the goals of this project and the guidance to follow them successfully. Our meetings were always very fruitful and his revision of my report very helpful.

I am grateful to Prof. Dr. Moira C. Norrie for giving me the opportunity to work on this semester project. Furthermore, I would like to thank Alex de Spindler for his help setting up and enhancing the iServer Web Service, Andreas Malär for correcting my report and giving me advice about its structure and Domenic Schröder for the inspiring discussions about Firefox extensions.

List of Figures

A
Development environment

This chapter describes some changes you can do to your system to make extension development in Firefox easier. The description below is intended to work with Firefox 1.5, but most parts should also work with Firefox 1.0.

The first section describes how to set up the development environment with a second Firefox instance installed on your system. The second section contains information about the file structure for an extension and the development cycle. At the end of this section we discuss how to create the JavaScript documentation for an extension.

A.1 Set up environment

To prevent your daily used Firefox from potential crashes due to bugs in your extension you have to create a separate profile. To do so, start the Firefox profilemanager by typing `./firefox -profilemanager` (this assumes that you are in the firefox directory) in your command line and then create a new profile.

You can run two instances of Firefox using separate profiles if you set `MOZ_NO_REMOTE` environment variable to 1. For example, on Windows you can use the following bat file to run Firefox with development profile, whether ”normal” Firefox is already running or not. (Assuming your development profile is called ”dev”):

```
set MOZ_NO_REMOTE=1
```

```
firefox -p dev
```

Change the window icon

If you use a separate installation of Firefox for development you can change Firefox's default window icon to any icon you want. This makes it easier to distinguish the development Firefox instance from the default Firefox. You change the icon by following these steps:

1. Go to the folder you installed Firefox in (for example `C:\Program Files\Mozilla Firefox_dev\`) and then go to the subfolder chrome.
2. While in chrome, create a new subfolder called icons, then go to that folder and create yet another subfolder called default. The full path to this folder could be for example: `C:\Program Files\Mozilla Firefox_dev\chrome\icons\default\`
3. Choose the icon you want to use and then place it in this folder and rename it to `main-window.ico`.

In addition to the main window, you can also change the icon on the Bookmark Manager and JavaScript Console in the same way. The icon names are `bookmark-window.ico` and `jsconsoleWindow.ico`, respectively.

Set development preferences

Before you start developing, you should set some preferences to make life easier. You access them by typing `about:config` in your address bar.

- **javascript.options.showInConsole = true.** Logs errors in chrome files to the JavaScript Console to make debugging easier.
- **nglayout.debug.disable_xul_cache = true.** Disables the XUL cache so that changes do not require a restart.
- **browser.dom.window.dump.enabled = true.** Enables the use of the dump() statement to print to the standard console. (The application must be started using the -console flag)
- **javascript.options.strict = true.** Enables strict Javascript warnings in the JavaScript Console. Note that since many people have this setting turned off when developing, you will see lots of warnings for problems with their code in addition to warnings for your own extension.

Install development extensions

The following developer extensions are also very useful (The first two come with the standard installation):

- The DOM Inspector is useful when you are trying to find the id of XUL elements which you wish to modify .
- Not really a developer extension, but the JavaScript Console can be surprisingly useful. Open it every time you have made changes to your JS files; it serves as a handy syntax checker (but make sure you have set the preferences listed above).
- For a more advanced JavaScript aid, you can use Venkman[1], the Mozilla-based JavaScript debugger.
- Extension developer's extension[2] for Firefox/Thunderbird/Mozilla. Includes these tools:
    - Reload all chrome: Reloads a changed extension without restart
    - JS Shell: Execute statements from main application window. Features tab completion!
    - JS Environment: Run code snippets
    - XUL Editor: XUL editor with real-time preview
    - HTML Editor: HTML editor with real-time preview
    - Extension Builder: A tool for editing install.rdf, packaging and installing your extension, still under development
    - As a bonus, it can invoke the debugging preferences listed above with a single menu click.
- Console Filter[3] helps you to find the relevant errors in the JavaScript Console.
- DebugLogger[4] is a developer extension for Firefox that provides a better alternative to using the dump statement. It breaks up debug statements to be unique per extension or project and lets you view them independently in an empty console.

[1]http://www.mozilla.org/projects/venkman/
[2]http://ted.mielczarek.org/code/mozilla/extensiondev/
[3]http://forums.mozillazine.org/viewtopic.php?t=264146
[4]http://mozmonkey.com/debuglogger/

A.2 Development

For easier development install the extension in a non packaged structure, because otherwise you have to make a new package each time you change something. How to work with an unpackaged extension is described in this section. Also the development cycle and the creation of the JavaScript documentation.

File structure of the iServext extension

You have to use two different file structures for the iServext. One for development (found in `\xhtml\iserver\src\`) und one for packaging (found in `CD:\iservext_pack\`). The install.rdf and chrome.manifest files contain the information needed for registering the extension. In the preferences.js the default preferences are stored. The content folder holds all the JavaScript and XUL files needed in the iServext. The iservext.dtd contains the entities for English. Other languages could be added in the respective folder. In the skin folder the CSS files and all the pictures are stored.

Developing

```
helloworld/
  chrome.manifest
  install.rdf
  defaults/
    preferences/
      preferences.js
  content/
    overlay.js
    overlay.xul
    ...
  locale/
    en-US/
      iservext.dtd
      iservext.properties
  skin/
    iservext.css
    ...
```

Packaging

```
iservext.xpi/
  chrome.manifest
  install.rdf
  defaults/
    preferences/
      preferences.js
  chrome/
    iservext.jar
      content/
        overlay.js
        overlay.xul
        ...
      locale/
        en-US/
          iservext.dtd
          iservext.properties
      skin/
        iservext.css
        ...
```

The `chrome.manifest` is also different for development and packaging. The `chrome.manifest` for packaging is printed below and contains an extra `jar:chrome/iservext.jar!`. This points the extension manager to the jar file in the chrome folder.

```
overlay chrome://browser/content/browser.xul...
           ...chrome://iservext/content/iservext.xul

content iservext  jar:chrome/iservext.jar!/content/

locale  iservext  en-US jar:chrome/iservext.jar!/locale/en-US/

skin    iservext  classic/1.0 jar:chrome/iservext.jar!/skin/
style   chrome://global/content/customizeToolbar.xul
           ...chrome://iservext/skin/iservext.css
style   chrome://browser/content/browser.xul
           ...chrome://iservext/skin/iservext.css
```

The chrome.manifest for developing is printed below and does not need any extra information, because the files can be accessed directly.

```
overlay chrome://browser/content/browser.xul...
           ...chrome://iservext/content/iservext.xul

content iservext    content/

locale  iservext    en-US   locale/en-US/

skin    iservext    classic/1.0 skin/
style   chrome://global/content/customizeToolbar.xul...
           ...chrome://iservext/skin/iservext.css
style   chrome://browser/content/browser.xul...
           ...chrome://iservext/skin/iservext.css
```

For the packaging version the content, locale and skin folder are put into iservext.jar which is a simple ZIP file. All the components are then added to iservext.xpi which is a ZIP file, too. The iservext.xpi can be installed in any compatible Mozilla Firefox.

Registering your extension in the Extension Manager

As described above you can use an unpackaged version of your extension while developing. If you do so, you have to register your extension manually. Proceed as follows:

The iservext's unique GUID is `{0d70c0bb-05a2-410d-b3f4-a91f3270a0bf}`. We assume your extension is in the folder `C:\develop\iservext`.

To register it, you put this line `C:\develop\iservext` in a textfile named `{0d70c0bb-05a2-410d-b3f4-a91f3270a0bf}` and store it in `\profile_folder\extensions\`.

Development cycle

Once you have registered your extension following the steps above, developing your extension is quite easy. If you have set the development preferences, your development cycle will be like this:

1. Edit your extension files.
2. Reopen the window, that modified files apply to, or use the Reload chrome feature of the Extension Developer Extension.
    - If you changed chrome.manifest, you will have to restart.
    - If you changed install.rdf, you need to touch the extension folder specified in your ”GUID” file (update its Last modified time) and restart.

Further information about extension development can be found at:
`http://kb.mozillazine.org/Extension_development`

Creating the JavaScript documentation

The JavaScript documentation we created with a slightly modified version of JSDoc[5]. It is available at `CD:\jsdoc\`. All you need is a Perl runtime environment on your computer with the HTML:Template module installed. For Windows we recommend ActivePerl[6] and the HTML:Template can be copied form the `CD:\jsdoc\HTML\` into `PERL_Home\lib\HTML\`. The HTML:Template module is also available online[7].

To easily create an new JavaScript documentation for iServext, you can use the `iservext_jsdoc.bat` which creates it from the files in the source folder. It is created in the `destination\iservext\` folder. You can edit the bat file to adapt the documentation to your needs.

[5]http://jsdoc.sourceforge.net/
[6]http://www.activeperl.com/
[7]http://html-template.sourceforge.net/

B User manual

In this chapter we provide a user manual for the installation of the iServer Web Service and the iServext. This user manual also covers the usage of the iServext.

B.1 Installation

First, we define the prerequisites for the installation. In the next section the main steps of setting up the the iServer Web Service are described and illustrated. In the following section we the describe the installation of the iServext.

For the next steps the following prerequisites are required:

- The Tomcat Webserver[1] is installed on your computer. We used a standalone Tomcat 4.1. All other version which are compatible with Axis Web Services[2] can also be used.
- Firefox 1.0+ or newer is installed on your computer. A tested version of the Firefox 1.0+ can be found in the folder `\xhtml\iservext\software\`
- Your Firefox has the XPointerLib version 0.2.2.1 installed. It can be found in the folder `\xhtml\iservext\deploy\`

If these requirements are met, you can start with the installation of the iServer Webservice.

[1]Information and download: http://jakarta.apache.org/tomcat/

[2]Information and download: http://ws.apache.org/axis/

B.1.1 Installing the iServer Webs Service

The installation of iServer Web Service consists mainly of copying the right files to the right place. In the following `TOMCAT_HOME` is assumed to be the directory where your Tomcat Server has been installed. First you have to copy the Axis libraries as follows:

- Copy `axis.jar`, `commons-discovery.jar`, `commons-logging.jar` and the `wsdl4j.jar` to the `TOMCAT_HOME\shared\lib\` directory on the application server.
- Copy log4j-1.2.4.jar to the `TOMCAT_HOME\common\lib\` directory on the application
- Copy jaxrpc.jar and the saaj.jar to the `TOMCAT_HOME\common\endorsed\` directory on the application server.

These files can be found in the folder `\iserverp2p\lib\` or `CD:\axis_for_building\lib\`

In the second step the iServer Web Service directory has to be created in, or copied to `TOMCAT_HOME\webapps\`. The directory structure should be as shown in Figure B.1. The `lib` folder should contain: `iserver.jar`, `jdom.jar`, `paperpp.jar`, `sigtec.jar`, `xdatabase.jar`, `xhtml.jar` and the `xima.jar`. The `paperpp.jar` has to be replaced by the `ipaper.jar` in a updated version.

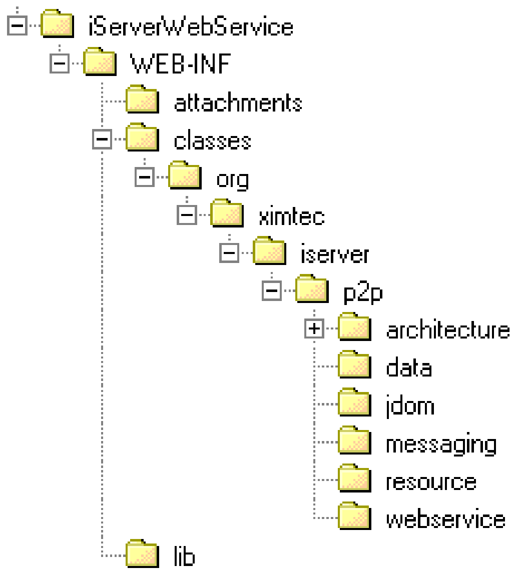


Figure B.1: Directory structure for iServer Web Service

The classes folder contains the class files of the iServer P2P project in an adjusted version. All these files can be found in the folder `CD:\tomcat_webapps\` and the jar files are also in the deploy folder of the corresponding projects.

In the adjusted version the following was altered: The `XMLElementFactoryInitialiser` class and the `JdomOMInstanceElement` class in the `org.ximtec.iserver.p2p.jdom` package and the whole `org.ximtec.iserver.p2p.webservice` package, too. Details about the changes in this classes are given in the last subsection of this section.

Before the iServer Web Service can be used it has to be deployed in the Tomcat Webserver. Therefore you have to make sure the `web.xml` exist in the `WEB-INF` folder and the Tomcat is restarted. The `web.xml` should be already if you copied the `iServerWebService` folder from `CD:\tomcat_webapps\` otherwise it can be copied from there. Afterwards you can deploy the service by using the command below. In the classpath the following jar files have to be made avaible: `axis.jar`, `commons-discovery.jar`, `commons-logging.jar`, `jaxrpc.jar`, `log4j-1.2.8.jar`, `saaj.jar` and `wsdl4j.jar`.

```
java -cp \%cp\%;. org.apache.axis.client.AdminClient...
                                        ...deploy.wsdd
-l http://localhost:8080/iServerWebService/services/...
                                        ...AdminService
```

The `deploy.wsdd` file can be found in the `CD:\axis_for_building\classes\org\ximtec\iserver\p2p\webservice\` folder or generated as described in the last subsection. For easier deployment of the Web Service you can use the `deploy.bat` which lies in the same folder as the the `deploy.wsdd`. It is assumed that your Tomcat path is `http://localhost:8080` otherwise this has to be adjusted. If call the deployment command it gives the following output if successful.

```
Processing file deploy.wsdd
<Admin>Done processing</Admin>
```

If it does not work you should recheck your classpath if the necessary files are included. After you deployed the Web Service you can restart your Tomcat Webserver and open the URL `http://localhost:8080/iServerWebService/` services/IServerWeb Service and you should see the massage "Hi there, this is an AXIS service!". If see this massage your iServer Web Service is installed successfully.

After this first start of the Web Service a `config.properties` file is created in the directory in which your Tomcat Webserver ist running. This can be your `TOMCAT_HOME\bin` or if you use a standalone Tomcat Webserver your Tomcat startmenu folder for example. After you have setup your database you have to adapt this file like we describe in the next section.

Set-up of the Database

For set-up you can use the files provided in the \xhtml\ folder. You can copy the xhtmlDatabaseEmpty.dml and rename it to xhtmlDatabase.dml. In the xhtmlDatabase.xml file you can define the users and possible predefined links you want to use. After changing the XML file you have to run the dml_update.bat and then the OMS_replace.bat. They will write the information from the XML file into the database. The database is then ready for use.

Before you can use it you have the have to set the correct keys for the database name and location in the config.properties file. Possible entries could look like this:

```
<entry key="DATABASE_NAME">xhtmlDatabase</entry>
<entry key="DATABASE_LOCATION">C:/develop/xhtml</entry>
```

After you have changed the file you restart the Tomcat Webserver and the iServer Web Service is ready for usage.

Adding new methods

This section describes how you can add new methods to the iServer Webservice. If you do not intend to do development of the iServext you can skip this section.

If you want to add a new method to the Web Service you can define it in the IServerWebService class and then compile the class. Afterwards you can copy the IServeWebService.class file into a folder you copied from CD:\axis_for_building\classes\org\ximtec\iserver\p2p\webservice\ (see Figure B.2).
Then you can create new java files for the Web Service by using the JAVA2WSDL and WSDL2JAVA methods. They are called as described below.

```
java -cp \%cp\%;. org.apache.axis.wsdl.Java2WSDL-o deploy.wsdl
-l "http://localhost:8080/iServerWebService/services/...
                                        ...IServerWebService"
-n "urn:IServerWebService"
-p"org.ximtec.iserver.p2p.webservice" "urn:IServerWebService"
org.ximtec.iserver.p2p.webservice.IServerWeb Service
```

```
java -cp \%cp\%;. org.apache.axis.wsdl.WSDL2Java -o .
-d Session -s -S true
-Nurn:IServerWeb Serviceorg.ximtec.iserver.p2p.WebService...
                                        ...deploy.wsdl
```

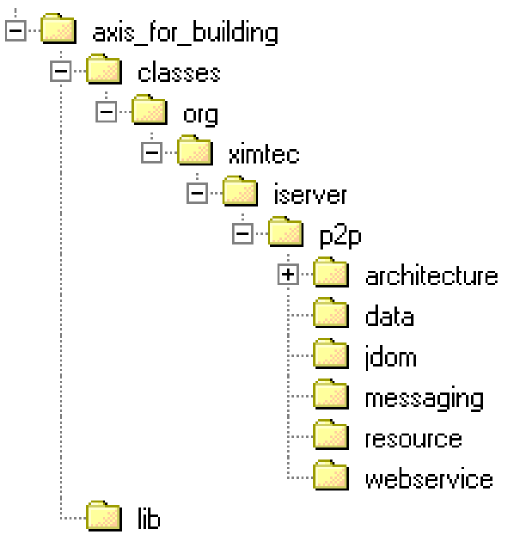


Figure B.2: Directory structure for the axis building folders

The classpath has to contain all the axis files as it does for deploying the Web Service. For these two steps you can use the bat files in the `CD:\axis_for_building\classes\` folder. After invoking these methods a new `deploy.wsdl` file and the new java files for the Web Service will be generated.

These files you can copy back into your project folder and recompile them. Before you compile them, you have to implement the method you have defined in the `IServiceWebService` class. This has to be done in the `IServerWebServiceSoapBindingImpl` class. Because all the methods in this file are newly created it is easier to rename it before coping it to the project folder and then just copy the new method into the original file.

When the method is implemented and the all Web Service classes are compiled you can copy them into the iServer Web Service folder in your Tomcat directory. Afterwards you have to restart the Tomcat Webserver.

If you want to add a new datatype in the Web Service you have to do this in two files. They both are from the `org.ximtec.iserver.p2p.jdom` package. In the `XMLElementFactoryInitialiser` class the new type has to be registered. This is done by adding lines like the following which are for the XHTMLSelector. The packages which are used have to be made available in the project and also the Tomcat lib folder.

```
XMLElementFactory.register(
    org.ximtec.xhtml.core.XHTMLSelector.class.getName(),
    org.ximtec.xhtml.jdom.XMLSchema.XHTML_SELECTOR,
    JdomOMInstanceElement.class.getName());
```

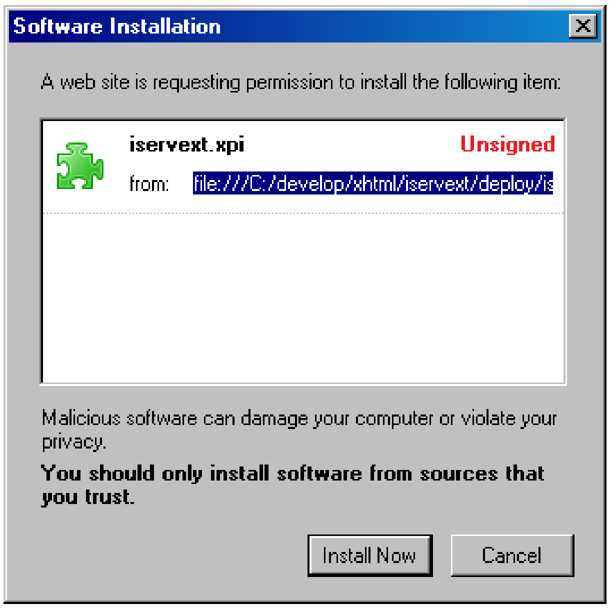
In the `JdomOMInstanceElement` class a new constructor for the new datatype has to be added. Like in the example for the XHTMLSelector below.

```
public JdomOMInstanceElement(XHTMLSelector xhtmlSelector) {
    this((OMInstance)xhtmlSelector);
    }
```

After you added the code you have to compile the package and copy it into the Tomcat `iServerWebService` folder and restart the Tomcat Webserver.

B.1.2 Installing the iServext

The iServext can be installed as any other Firefox extension. You can open a the `iservext.xpi` file which is located in the `\xhtml\iservext\deploy\` folder. A window like shown in Figure B.3 will pop up and you just have to click Install Now and after a restart of your Firefox, the iServext will be installed.


Software Installation

A web site is requesting permission to install the following item:

iservext.xpi Unsigned

from: file:///C:/develop/xhtml/iservext/deploy/is

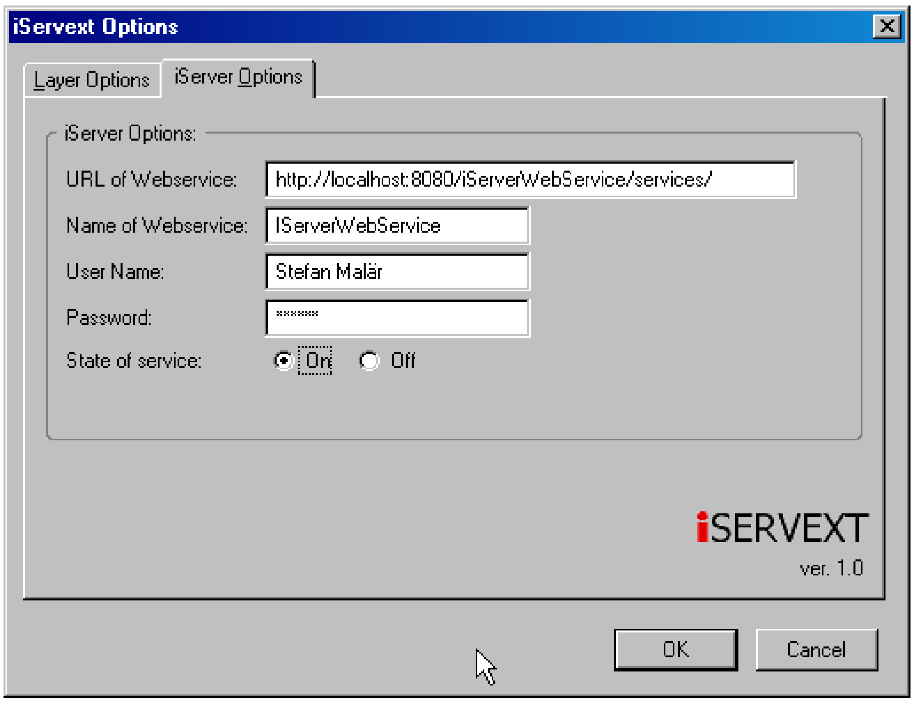
Malicious software can damage your computer or violate your privacy.

You should only install software from sources that you trust.

Figure B.3: Extension installation window

Then you have to set the iServer Web Service preference. You can do this if you open Extensions in the Tools menu. There you have to select the iServext and click on options. The preference window will show and you can select the iServer Options tab as shown in Figure B.4. There you have to set the URL of Web Service and the name of it. As in our example http://localhost:8080/iServerWebService/services/ and IServerWebService respectively. You also have to insert a user name which has to be defined in your database with help of the

XML file as described above. The password is not needed at the moment and you can also turn the service on and off. For turning the service on and off you can also add an icon in your toolbar. This is done by right clicking the toolbar and select customize. Then you can drag and drop the iServext icon into your toolbar.


iServext Options

Layer Options | iServer Options

iServer Options:

URL of Webservice: http://localhost:8080/iServerWebService/services/

Name of Webservice: IServerWebService

User Name: Stefan Malär

Password: ******

State of service: [✓ On] Off

Figure B.4: iServer options preference window

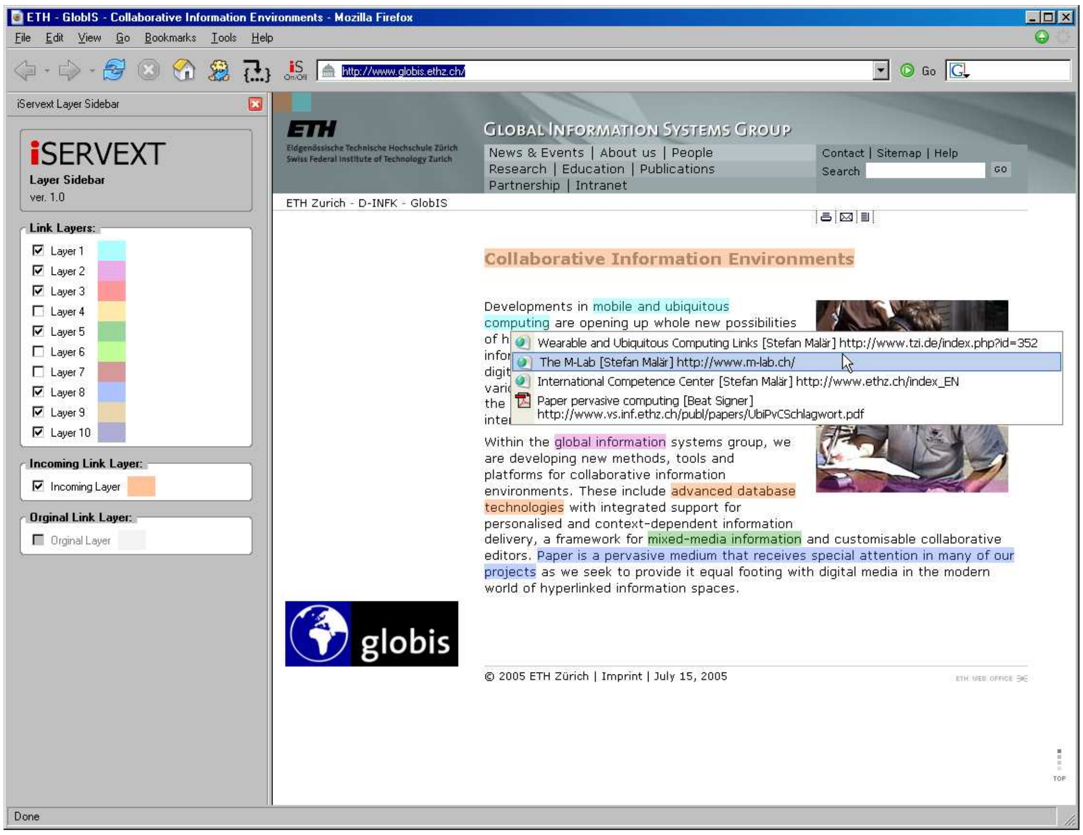
After you performed all these steps, the XHTML plug-in for iServer, the iServext is ready for use.

B.2 Usage of the iServext

The iServext can be used for browsing and authoring iServer XHTML links. In the next section we cover how to set the preference and use the layer sidebar. In second section we discuss the authoring of new links.

B.2.1 Browsing

For browsing you can use the iServext layer sidebar. It can be displayed by selecting iServext layer in the sidebar choice of the View menu or by simply pressing the keys `ctrl + Q`. The layer sidebar allows you to deselect layers which then are not displayed anymore. You can see the color of the different layers, too.


ETH - GlobIS - Collaborative Information Environments - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.globis.ethz.ch/
Go
iServext Layer Sidebar
iSERVEXT
Layer Sidebar
ver. 1.0
Done

Link Layers:

☑ Layer 1	🟦
☑ Layer 2	🟪
☑ Layer 3	🟥
☐ Layer 4	🟨
☑ Layer 5	🟩
☐ Layer 6	🟩
☐ Layer 7	🟥
☑ Layer 8	🟦
☑ Layer 9	🟫
☑ Layer 10	🟪

Incoming Link Layer:

☑ Incoming Layer 🟧

Orginal Link Layer:

☐ Orginal Layer

GLOBAL INFORMATION SYSTEMS GROUP

News & Events | About us | People
Research | Education | Publications
Partnership | Intranet
Contact | Sitemap | Help
Search GO

ETH Zurich - D-INFK - GlobIS

Collaborative Information Environments

Developments in mobile and ubiquitous computing are opening up whole new possibilities of h… infor… digit… vari… the… inte…

- Wearable and Ubiquitous Computing Links [Stefan Malär] http://www.tzi.de/index.php?id=352
- The M-Lab [Stefan Malär] http://www.m-lab.ch/
- International Competence Center [Stefan Malär] http://www.ethz.ch/index_EN
- Paper pervasive computing [Beat Signer] http://www.vs.inf.ethz.ch/publ/papers/UbiPvCSchlagwort.pdf

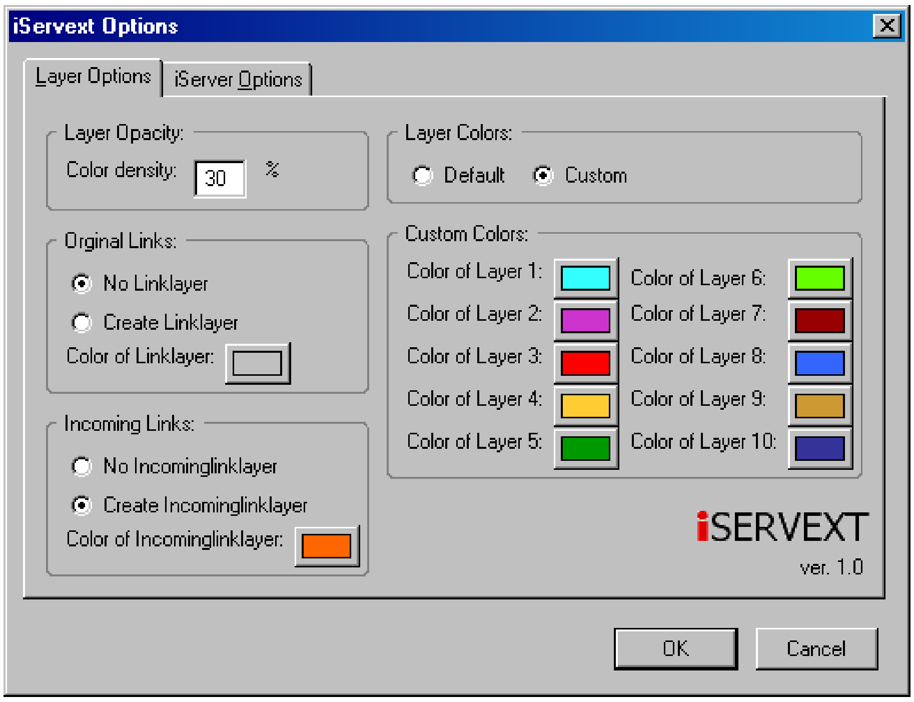
Within the global information systems group, we are developing new methods, tools and platforms for collaborative information environments. These include advanced database technologies with integrated support for personalised and context-dependent information delivery, a framework for mixed-media information and customisable collaborative editors. Paper is a pervasive medium that receives special attention in many of our projects as we seek to provide it equal footing with digital media in the modern world of hyperlinked information spaces.

© 2005 ETH Zürich | Imprint | July 15, 2005

Figure B.5: Screenshot of the Firefox with the layer sidebar

Whenever you load a new document the iServext checks whether iServer links exist for it. If there are links they are displayed as shown in Figure B.5. The links which link to or from the whole document are displayed in the upper left corner. If you click on the square, a list with all the links shows up. Links which link to or from an XHTML Selector are displayed in a colored layer over the selector area. If just one target is defined for a source, the mouse pointer changes to the iServext symbol and you follow the link directly by clicking on the layer. Otherwise, a list symbol is shown and the targets are displayed in a link list as shown in Figure B.5.

Furthermore, you have the possibility to set various preferences. You can do this if you open Extensions in the Tools menu. There you have to select the iServext and click on options. The preference window will show as in Figure B.6.



Layer Opacity:
Color density: 30 %

Layer Colors:
Default [✓ Custom]

Orginal Links:
[✓ No Linklayer]
Create Linklayer
Color of Linklayer: ⬜

Incoming Links:
No Incominglinklayer
[✓ Create Incominglinklayer]
Color of Incominglinklayer: 🟧

Custom Colors:

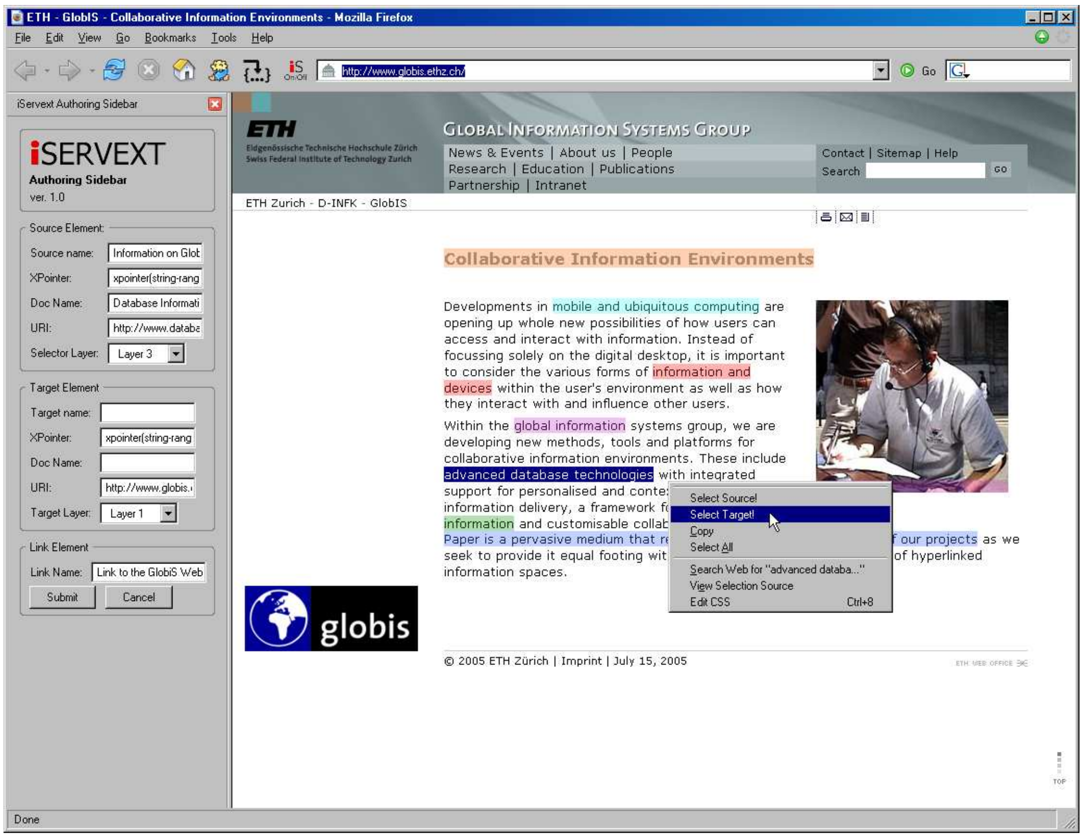
Color of Layer 1:	🟦	Color of Layer 6:	🟩
Color of Layer 2:	🟪	Color of Layer 7:	🟥
Color of Layer 3:	🟥	Color of Layer 8:	🟦
Color of Layer 4:	🟨	Color of Layer 9:	🟧
Color of Layer 5:	🟩	Color of Layer 10:	🟦

Figure B.6: Layer options preference window

In the first box on the left side you set the opacity of the layers. In the second box you choose if you want to replace the original links of a document by links on a layer and you can choose the layer color for this layer. The third box lets you display incoming links and define the color of the incoming link layer. On the right hand side you can choose if you want to use the default layer colors for the outgoing link layers or custom colors. The custom colors can be defined below for each layer.

B.2.2 Authoring of new links

For the authoring of new links the iServext offers the authoring sidebar. This sidebar can be displayed by selecting iServext authoring in the sidebar choice of the View menu or by simply pressing the keys `ctrl + Y`. Afterwards you can select a part of a document or the whole document with a right click of the mouse and pressing of select source or select target as shown in Figure B.7.


ETH - GlobIS - Collaborative Information Environments - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.globis.ethz.ch/
Go
iServext Authoring Sidebar
iSERVEXT
Authoring Sidebar
ver. 1.0
Submit
Cancel
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
GLOBAL INFORMATION SYSTEMS GROUP
News & Events | About us | People
Research | Education | Publications
Partnership | Intranet
Contact | Sitemap | Help
Search
GO
ETH Zurich - D-INFK - GlobIS
Select Source!
Select Target!
Copy
Select All
Search Web for "advanced databa..."
View Selection Source
Edit CSS Ctrl+8
globis
Done

Source Element:
Source name: Information on Glot
XPointer: xpointer(string-rang
Doc Name: Database Informati
URI: http://www.databa
Selector Layer: Layer 3

Target Element
Target name:
XPointer: xpointer(string-rang
Doc Name:
URI: http://www.globis.
Target Layer: Layer 1

Link Element
Link Name: Link to the GlobiS Web

Collaborative Information Environments

Developments in mobile and ubiquitous computing are opening up whole new possibilities of how users can access and interact with information. Instead of focussing solely on the digital desktop, it is important to consider the various forms of information and devices within the user's environment as well as how they interact with and influence other users.

Within the global information systems group, we are developing new methods, tools and platforms for collaborative information environments. These include advanced database technologies with integrated support for personalised and conte information delivery, a framework fo information and customisable collab Paper is a pervasive medium that re f our projects as we seek to provide it equal footing wit of hyperlinked information spaces.

© 2005 ETH Zürich | Imprint | July 15, 2005

Figure B.7: Screenshot of the Firefox with the authoring sidebar

The selection is then shown in the appropriate fields in the sidebar. You also have the possibility to give a name to the document and the selector and you have to give a name to the link. If you want to link to a different document you just open a new tab and load it there and then select the target. After you have finished you can submit the link and a confirmation will show up if the creation was successful.

For the source and the target you can specify a layer. In case the selector already exists on a layer you have to move it to another layer. The different layers for the targets are not displayed in the browser in favor of clarity.

The iServer just uses a runtime version of the link database. If you want your newly created links persistent you have to do this explicitly. You can do it by clicking iServext SaveDB! in the tools menu. The current state of the database will then be committed and saved over the latest version.

C Reference to future work

In this chapter we will give some reference for future work on the limitations and the enhancement of the iServext. We discuss some ideas of possible approaches and point out the appropriate code locations.

C.1 Remove limitations

In this section we present how some of the current limitations could be removed. The limitation to XHTML links is discussed in the next section about new features which would in this case be the support of other links.

Support of frames

Websites with frames are currently not supported because we only insert the link information into the main page which does not work if this page is the parent of other pages in frames.

We suggest that these limitations are removed by the enhancement of the `newPageLoad()` method in the `overlay.js` file. The `aDoc` object which represents the current document could be examined if it has frames as children. In the case it does the link information for each of them could be called from the link database and implemented.

Handling of Selector intersections

If a `Selector` intersects with another one the correct visualization of both cannot be provided. One approach would be to prohibited the use of such selectors at the time of link creation. Another approach could be the handling of them at the time of the link integration and visualization.

For the second approach a redesign of the current link visualization and accessing model would have to be done. Instead of accessing the links through the layer behavior they could be accessed through a map of coordinates of the layers and the current mouse position. Therefore, all the layer coordinates of the links would have to be written into a kind of table and on mouse click the position of the mouse pointer had to be checked and the appropriate action chosen. If this would be the best method for doing this would have to be reconsidered more exhaustively.

Removing the time delay

The problem with the time delay is an issue of the iServer Web Service. As soon as it is fixed, the time delay in the iServext can be removed. To remove it the `setTimeout()` method shown below can be replaced by just the `getIncomingLinks()` method in the `newPageLoad()` method in the `overlay.js` file.

```
//if incoming links are displayed add the layer and get them
if(getPrefBranch().getBoolPref('incominglayer')){
    addIncomingLayer(doc);
    setTimeout("getIncomingLinks();",5000);
}
```

The `getIncomingLinks()` can then be called with the document object as an argument like the `getOutgoingLinks(doc)` method. It has to be adapted to handle the argument. The global variable `gDoc` in the `getLinks.js` file could then be removed as well.

C.2 New features

Some of the possible new features of the iServext are discussed in this section. In some cases they also could remove some current limitations.

Dynamic layers

Instead of the ten fix layers, dynamic layers could be used. The layer information is stored in the link database and also provided in the link information.

In the `inserLinks()` method this information is already extracted as shown here:

```
var layerName = sources[q].getElementsByTagName('layer')[0].
        getElementsByTagName('name')[0].firstChild.nodeValue;
```

This information could be used to dynamically create the appropriate CSS classes and insert them with the help of a modified `addCSS()` method which is found in the `css.js` file. The `insertWholeLayer()` would not have to be changed since only one layer for incoming links and one for the outgoing is used.

Beforehand it would advisable to consider if such dynamic layers also could be used with iServer P2P or if general adaptations to current layer usage would have to be made.

Support of other links

The support of other link type surly has to be a goal of the future development of the iServext. Only then the whole advantages of the iServer framework can be fully exploited.

The integration of other targets then XHTML documents could be done in the `insertLinks()` method found in the `getLinks.js` file. The extraction of the link information could be done exactly the same way it is done for the XHTML links. The actions for the onClick event would have to be set appropriate to the target media. How the target media would be displayed depends mainly on the state of the tools available for the iServer framework at the time of implementation. For an external application maybe a MIME type would have to be defined in Firefox.

For the visualization of different target media, the mouse pointer could be used with different symbols. The symbols could be chosen similar the code below used for PDFs or with the help additional link information.

```
//If the resource is a pdf set the icon for it
if(aTargetUri[w].indexOf('.pdf') >= 0){
    menuTextString += 'cursorMenu_'+aLinkNr+'.items.item' +w+
    '.showIcon("icon_pdf", ""); \n';
}
else{
    menuTextString += 'cursorMenu_'+aLinkNr+'.items.item' +w+
    '.showIcon("icon1", ""); \n';
}
```

The authoring of links other than XHTML links should be done in external tool for example the authoring tool. The possible collaboration with it is described in the next subsection.

Collaboration with the authoring tool

The collaboration with the authoring tool could be through SOAP if it is going to offer a Web Service or through socket communication. If SOAP is used the current methods in the `soap.js` file could be adapted. Information about socket communication within a Firefox can be found on XULPlanet[1]. We cannot provide more detailed information at the moment because the final API of the authoring tool is not available yet.

Localization

The localization of the iServext is very simple due to the use of dtd and properties files. The `iservext.dtd` and the `iservext.properties` files both found in the `chrome://iservext/locale/en-US/` folder have to be translated and copied for example into the `chrome://iservext/locale/de-DE/` folder for the German translation. In the chrome.manifest file a line as shown below has to be added.

```
locale  iservext    de-DE   locale/de-DE/
```

[1]http://www.xulplanet.com/references/xpcomref/group_Network.html#Sockets

Bibliography

[1] Kenneth M. Anderson. Integrating open hypermedia systems with the world wide web. In *HYPERTEXT '97: Proceedings of the eighth ACM conference on Hypertext*, pages 157–166. ACM Press, 1997. Southampton, United Kingdom.

[2] Kenneth M. Anderson, Richard N. Taylor, and Jr. E. James Whitehead. Chimera: hypertext for heterogeneous software environments. In *ECHT '94: Proceedings of the 1994 ACM European conference on Hypermedia technology*, pages 94–107. ACM Press, 1994. Edinburgh, Scotland.

[3] Anders Berglund, Scott Boag, Don Chamberlin, Mary F. Fernández, Michael Kay, Jonathan Robie, and Jérôme Siméon. Xml path language (xpath) 2.0. W3C Working Draft 04 April 2005 http://www.w3.org/TR/xpath20/.

[4] Scott Boag, Don Chamberlin, Mary F. Fernández, Daniela Florescu, Jonathan Robie, and Jérôme Siméon. Xquery 1.0: An xml query language. W3C Working Draft 04 April 2005 http://www.w3.org/TR/xquery/.

[5] Niels Olof Bouvin. Unifying strategies for web augmentation. In *HYPERTEXT 1999, Proceedings of the 10th ACM Conference on Hypertext and Hypermedia: returning to our diverse roots*, pages 91–100, Darmstadt, Germany, 1999. ACM.

[6] Niels Olof Bouvin. *Augmenting the web through open hypermedia*. Phd, University of Aarhus, 2000.

[7] L. A. Carr, W. Hall, and S. Hitchcock. Link services or link agents? In *HYPERTEXT '98: Proceedings of the ninth ACM conference on Hypertext and hypermedia : links, objects, time and space—structure in hypermedia systems*, pages 113–122. ACM Press, 1998. Pittsburgh, Pennsylvania, United States.

[8] L. A. Carr, D. De Roure, W. Hall, and G. Hill. Implementing an open link service for the world wide web. *World Wide Web*, 1(2):61–71, 1998.

[9] Leslie Carr, David De Roure, Wendy Hall, and Gary Hill. The distributed link service: A tool for publishers, authors and readers. In *Proceeding of the 4th International World Wide Web 95 Conference*, Boston, USA, 1995.

[10] Leslie Carr, Gary Hill, David de Roure, Wendy Hall, and Hugh Davis. Open information services. In *Proceedings of the fifth international World Wide Web conference on Computer networks and ISDN systems*, pages 1027–1036. Elsevier Science Publishers B. V., 1996. Paris, France.

[11] Bent Guldbjerg Christensen and Frank Allan Hansen. Xlink - linking the web and open hypermedia. In David Millard, Jrg M. Haake, and Siegfried Reich, editors, *Proceedings of the 2002 Workshop on Open Hypermedia Systems*, College Park, Maryland, 2002.

[12] Bent Guldbjerg Christensen, Frank Allan Hansen, and Niels Olof Bouvin. Xspect: Bridging open hypermedia and xlink. In *Proceedings of the Twelfth International World Wide Web Conference*, pages 490–499, Budapest, Hungary, 2003. ACM Press.

[13] Paolo Ciancarini, Federico Folli, Davide Rossi, and Fabio Vitali. Xlinkproxy: External linkbases with xlink. In Ethan V. Munson, Richard Furuta, and Jonathan I. Maletic, editors, *Proceedings of the 2002 ACM Symposium on Document Engineering*, pages 57–65, McLean, Virginia, 2002. ACM Press.

[14] James Clark and Steven J. DeRose. Xml path language (xpath) version 1.0. W3C Recommendation 16 November 1999 http://www.w3.org/TR/xpath.

[15] Alexandre de Spindler. *Distributed Collaborative Information Environment based on iServer*. Diploma thesis, Swiss Federal Institute of Technology (ETHZ), 2005.

[16] Steven J. DeRose. Xml linking. *ACM Comput. Surv.*, 31(4es):21, 1999.

[17] Steven J. DeRose, James Clark, and David Orchard. Xml linking language (xlink) version 1.0. W3C Recommendation 27 June 2001 http://www.w3.org/TR/xlink/.

[18] Steven J. DeRose, Ron Jr. Daniel, Eve Maler, and Jonathan Marsh. Xpointer xmlns() scheme. W3C Recommendation 25 March 2003 http://www.w3.org/TR/xptr-xmlns/.

[19] Steven J. DeRose, Eve Maler, and Ron Jr. Daniel. Xpointer xpointer() scheme. W3C Working Draft 19 December 2002 http://www.w3.org/TR/xptr-xpointer/.

[20] Anthony J. Duhig. Separating links from content using xml, xlink and xpointer. In *Proceedings of XML Europe 2001*, Berlin, Germany, 2001.

[21] Kaj Grønbæk, Niels Olof Bouvin, and Lennert Sloth. Designing dexter-based hypermedia services for the world wide web, 1997. Southampton, United Kingdom.

[22] Kaj Grønbæk, Lennert Sloth, and Niels Olof Bouvin. Open hypermedia as user controlled meta data for the web. In *Proceedings of the 9th international World Wide Web conference on Computer networks : the international journal of computer and telecommunications netowrking*, pages 553–566. North-Holland Publishing Co., 2000. Amsterdam, The Netherlands.

[23] Kaj Grønbæk, Lennert Sloth, and Peter Ørbæk. Webvise: browser and proxy support for open hypermedia structuring mechanisms on the world wide web. In *WWW '99: Proceeding of the eighth international conference on World Wide Web*, pages 1331–1345. Elsevier North-Holland, Inc., 1999. Toronto, Canada.

[24] Paul Grosso, Eve Maler, Jonathan Marsh, and Norman Walsh. Xpointer element() scheme. W3C Recommendation 25 March 2003 http://www.w3.org/TR/xptr-element/.

[25] Paul Grosso, Eve Maler, Jonathan Marsh, and Norman Walsh. Xpointer framework. W3C Recommendation 25 March 2003 http://www.w3.org/TR/xptr-framework/.

[26] Wendi Hall, Hugh Davis, and Gerard Hutchings. *Rethinking hypermedia the microcosm approach*. Kluwer, Boston etc., 1996. by Wendi Hall, Hugh Davis, Gerard Hutchings 25 cm Ill.

[27] W3C and INRIA. Amaya project. http://www.w3.org/Amaya/.

[28] Michael Kay. Xsl transformations (xslt) version 2.0. W3C Working Draft 4 April 2005 http://www.w3.org/TR/xslt20/.

[29] Theodorich Kopetzky and Max Mühlhäuser. Visual preview for link traversal on the world wide web. *Computer Networks: The International Journal of Computer and Telecommunications Networking (Proceeding of the eighth international conference on World Wide Web)*, 31(11-16):1525–1532, 1999.

[30] Hermann Maurer. *HyperWave the next generation Web solution*. Addison-Wesley, Harlow, England etc., 1996. Hermann Maurer; [foreword by Robert Cailliau] Ill. 1 CD-ROM.

[31] Débora C. Muchaluat-Saade, Rogério F. Rodrigues, and Luiz Fernando G. Soares. Xconnector: Extending xlink to provide multimedia synchronization. In *Proceedings of the 2002 ACM symposium on Document engineering*, pages 49–56, McLean, Virginia, USA, 2002. ACM Press.

[32] Jakob Nielsen. Using link titles to help users predict where thea are going, 11. January 1998 1998. Availble at: http://www.useit.com/alertbox/980111.html.

[33] Hartmut Obendorf and Harald Weinreich. Comparing link marker visualization techniques - changes in reading behavior. In *Proceedings of 12th International World Wide Web Conference - WWW 2003*, Budapest, HUNGARY, 2003.

[34] Glenn Oberholzer and Erik Wilde. Openly accessing linkbases. Technical Report TIK-Report No. 134, Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology, January 2002.

[35] Simon Schulé. *XHMTL Plug-in for the iServer Architecture*. Diploma thesis, Swiss Federal Institute of Technology (ETHZ), 2004.

[36] Beat Signer and Moira C. Norrie. A framework for cross-media information management. In *Proceedings of EuroIMSA 2005, International Conference on Internet and Multimedia Systems and Applications*, Grindelwald, Switzerland, 2005.

[37] Fabio Vitali, Federico Folli, and Claudio Tasso. Two implementations of xpointer. In *HYPERTEXT '02: Proceedings of the thirteenth ACM conference on Hypertext and hypermedia table of contents*, pages 145–146, College Park, Maryland, USA, 2002. ACM Press.

[38] W3C. Annotea project. http://www.w3.org/2001/Annotea/.

[39] W3C. Cascading style sheets. http://www.w3.org/Style/CSS/.

[40] Harald Weinreich, Hartmut Obendorf, and Wienfried Lamersdorf. The look of the link - concepts for the user interfac of extended hyperlinks. In *HYPERTEXT 2001, Proceedings of the 12th ACM Conference on Hypertext and Hypermedia*, pages 19–28, University of Aarhus, rhus, Denmark, 2001. ACM.

[41] Harald Weinreich, Hartmut Obendorf, and Wienfried Lamersdorf. Hyperscout: Darstellung erweiterter typinformationen im world wide web - konzepte und auswirkungen. In Gerd Szwillus Jrgen Ziegler, editor, *Mensch und Computer 2003*, Berichte des German Chapter of the ACM, pages 155–164. B.G. Teubner Verlag Stuttgart, Leipzig, Wiesbaden, 2003.

[42] Harald Weinreich, Hartmut Obendorf, and Wienfried Lamersdorf. Hyperscout: Linkvorschau im world wide web. *i-com: Zeitschrift fr interaktive und kooperative Medien*, 1(3):4–12, 2003.

[43] Erik Wilde and David Lowe. *XPath, XLink, XPointer, and XML a practical guide to Web hyperlinking and transclusion.* Addison-Wesley, Boston, MA, 2003. Erik Wilde, David Lowe Ill.

[44] Polle T. Zellweger, Bay-Wei Chang, and Jock D. Mackinlay. Fluid links for informed and incremental link transitions. In *HYPERTEXT '98: Proceedings of the ninth ACM conference on Hypertext and hypermedia : links, objects, time and space—structure in hypermedia systems*, pages 50–57. ACM Press, 1998. Pittsburgh, Pennsylvania, United States.