IMPROVED NAVIGATION THROUGH EXTENDED XML LINKS

Anna Lekova
IUSI,
Bulgarian Academy of Sciences
Bl 2, Acad. Bonchev St., 1113
Sofia, Bulgaria
alekova@iusi.bas.bg

Olga De Troyer
Vrije Universiteit Brussels,
Department of Computer Science
10G731D Pleinlaan 2 B-1050
Brussels, Belgium
Olga.DeTroyer@vub.ac.be

Abstract

The paper describes a browser that processes XLink links in XML documents. It creates a navigation chunk of hyperlinks dynamically for a connected user's session through a server side programming. This advanced navigation provides web-based online services while referencing the electronic documents in use, both local and remote. To allow communicating this information through the web, an XML/DTD documents for the manner in which Uniform Resource Identifiers should be referenced, are developed. Web GUIs are generated that enable users to explore and interact with the XLink elements available in such documents. The browser supports a graphical query notation based on XLink links in Java Graphics 2D applets.

1. Introduction

The WWW is the most dynamic accumulation of information. Unfortunately it is also the most lacking in structure. Users need a way to dynamically integrate the information available on the Internet. The frequent use of hyperlinks in web sites causes people to loose track of where they are and what they are looking for. Online-guided tours should be generated when novices are getting acquainted with web sites as well as during sessions of learning and teaching at a distance. Dynamic navigation could be extremely helpful in the daily work of office workers - to allocate their online tasks and web services in a way in which they are performed.

HTML linking has its limits [8]. For one thing, URLs are limited to pointing at a single document. Furthermore, HTML links don't maintain any sense of history or relations between documents. XML links (xlinks) propose more powerful links between documents. The XML Linking Language (XLink) [8] working draft defines two types of links, simple links and extended links. XLink

provides a framework for creating both basic unidirectional links and more complex linking structures. It allows XML documents to associate metadata with a link. XLink link associates a set of resources. Resource is any addressable unit of information or service (files, images, documents, programs, and query results). More than one resource can be referenced in a single "hot-spot" in the text. An extended link is a link that associates an arbitrary number of resources. In general, the extended XLink elements could be considered as meaningful wrappers that provide a nest for resources and arcs trough the following attributes: Locator Attribute (href); Semantic Attributes (role, arcrole, title); Behavior Attributes (show, actuate); Traversal Attributes (label, from, to). Therefore an XLink document has the potential to describe how the information is being integrated over the Internet by identifying the possible XLink points. Extended links are very important for situations where the participating resources are read-only, or where the resources are in formats with no native support for embedded links (such as many multimedia formats). An important application of XLink is in hypermedia systems that have hyperlinks. A hyperlink is a link that is intended primarily for presentation to a human user. Using or following a link for any purpose is called traversing [8].

Unfortunately, Elliotte Harold [4] considers that at the time of February 2001, XLink is still undergoing significant development and modification. To make matter worse, only Mozilla and Netscape 6.0 have any support for XLink links, and that support is incomplete. There are no general-purpose applications that support arbitrary links. XLink can be used for a lot more than just hypertext connections and embedded images in documents. Thus, even when XLink are fully implemented in browsers, they may not always be in blue underlined text that you click to jump to another page. In the meantime it is safer to convert XML to HTML on the server using Extensible Stylesheet Language [7].

If the power of the above mentioned XLink proposal draft [8] is implemented in a client/server framework, the end-user will be facilitated in exploring the Web for any custom application that needs to establish connections between documents and parts of documents. Interactive web tools for XLink analysis and synthesis need to be developed. Information about the longest path that one can take to reach a destination node could inform us where and how the end-user might get lost. Querying the XML document could be an easy way how to restructure the XLink resources and arcs and also how to generate personalized guided tours for information or task allocation.

Servlets [1] are an effective replacement for CGI scripts. They provide a way to generate dynamic documents that is both easier to write and faster to run. Any kind of browser can send a request to a set of servlets on the HTTP server when extended Links are encountered in the XML document. In the present work an approach for XLink links traversing and displaying directly in a browser is proposed. That approach creates the effect of having a dynamic, virtual temporary XML browser that creates a semantic navigation chunks for a connected user’s session through a server side programming. A Java Servlet using XML Link Language Processor combined with Document Object Model (DOM) manipulates the XML document that serves as a data exchange format for referencing documents between clients on the I-net. The servlet generates string buffers of applet parameters and responds through Java applets to the client browser. All of the title attributes of the potential extended link are presented in that applet. The end-user can click over the graphical notation of the titles. A Navigation Servlet is responsible for traversing and presenting the requested xlinks using the "arc" and "locator" attributes. It results in an HTML page to the HTTP client, which defines the path that one can take to reach a destination node. At the top of such a page a chunk of hyperlinks for the particular navigation is presented. Moreover the present approach allows advanced personalization of the XLink document by dragging, clipping and matching the graphical notation of the extended xlinks. The HTTP server processes the query composed this way dynamically. A Query Java Servlet generates a new XML document with new information for the

XLink. A more detail description of the approach is given in section 3. Section 2 discusses related work. In section 4 the approach is illustrated with an example.

2. Related Work

The number of tools for processing XLink and XPointers is limited [6]. The **Fujitsu XLink Processor (**XLip) [2] can work with any XML processor or parser, which can create DOMs, on condition that an appropriate interface between the processors is implemented. XLip cannot be independently obtained at present. Empolis Ltd. announced **X2X**, the XML XLink Engine [9]. X2X allows for the creation, management and manipulation of links. X2X has an extensible architecture to allow resources to reside in any data repository. X2X stores all the link information within an ODBC/ JDBC enabled database. X2X allows for the retrieval of resources and can dynamically add the external link information without altering the original document/information. **Linkan XML-XSL-XLL** browser [5] is a simple application written in Java that allows a user to view XML documents with XSL stylesheets and XLL hyperlinking. It uses several third-party Java libraries, including the Docuverse DOM implementation, James Clark's XP parser and XT XSL processor, and the XML-DEV mailing list's SAX API. The **XLink Filter** [10] uses John Cowan's ParserFilter class to insert itself between the parser and the application. The XLink filter (derived from the ParserFilter) passes all SAX events on to the application, which can do its own processing of the XML document represented by the SAX events. The XLink Filter will leave behind a Link Collection, which can take requests from the Application regarding which elements contain links and what the targets and behaviors of those links should be. If you click on an active applet area, a menu will appear, indicating available links. Links are multi-directional, and each location on the map may correspond to many clickable areas and its links.
In comparison with the above methods our framework is easier to use, both from the designer's and the user's point of view, as it employs interactive web tools for traversing and xlinks managing in graphical mode presentation.

3. A Client/Server Framework for XLink

The presented software framework for XML links processing demonstrates how one can create a servlet(s) and how a client uses HTTP to communicate with the servlet(s) for XML links traversing *(Fig.1.).* The entire XML document is manipulated dynamically using the XLink combined with Document Object Model through a Java server side programming. A set of Java HTTP Servlets respond to the client's requests based on the text in the XML & XSL documents as well as to the interactive work in 2D Graphics applets. A small code fragments are used to show how the web pages are produced and processed. The modeling of XLink scheme is shown in XML format:

```
<with_extendedL  xlink:title="C_book" xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="extended">
        <NAME xlink:type="resource" xlink:label="C_book"/>
        <START_P xlink:type="locator" xlink:href="http://www.amazon.com/" xlink:label="StartPageCust" …
        <TARGET xlink:type="locator" xlink:href="http://images.amazon.com/...jpg" xlink:label="info_targetCust"
                xlink:title="has_title"/>                <TARGET …
        <CONNECTION xlink:type="arc" xlink:from="CustomerPageCust" xlink:to="info_targetCust"/> …
</with_extendedL>
```

as well as in interactive 2D applet (*Fig.2.b.*) in order to make XML more easily accessible to a wider user audience. Here is looking for ways to connect XML documents more directly to in – graphic objects. It would be much easier to write XML-enabled queries if the components of an XML

document are mapped to in-graphic objects that represent the xlinks' intended meaning according to its XML schema.
The Extensible Stylesheet Language (XSL) is used for content visualization. XML documents are transformed to HTML documents, using a standard that applies fixed style rules to the content of particular elements. In an XSL transformation, an XSL processor reads both an XML document and an XSL. In the present work a servlet is integrated into XSL document. Any kind of browser can send a request to an HTTP servlet when an extended XLink has been activated *(fig.2.a).*

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/XSL/Transform/1.0"
          xmlns="http://www.w3.org/TR/REC-html40" result-ns="">
 <xsl:template match="/">
  <HTML> <BODY>
    <xsl:for-each select="order/item/book">
   <FORM ACTION=" http://starpc8.vub.ac.be:8080/servlet/XLink? XML_Name "  METHOD="POST">
    <INPUT TYPE="SUBMIT" VALUE=" XlinkMap"/>
  </FORM>
</xsl:for-each></xsl:template></xsl:stylesheet>
```

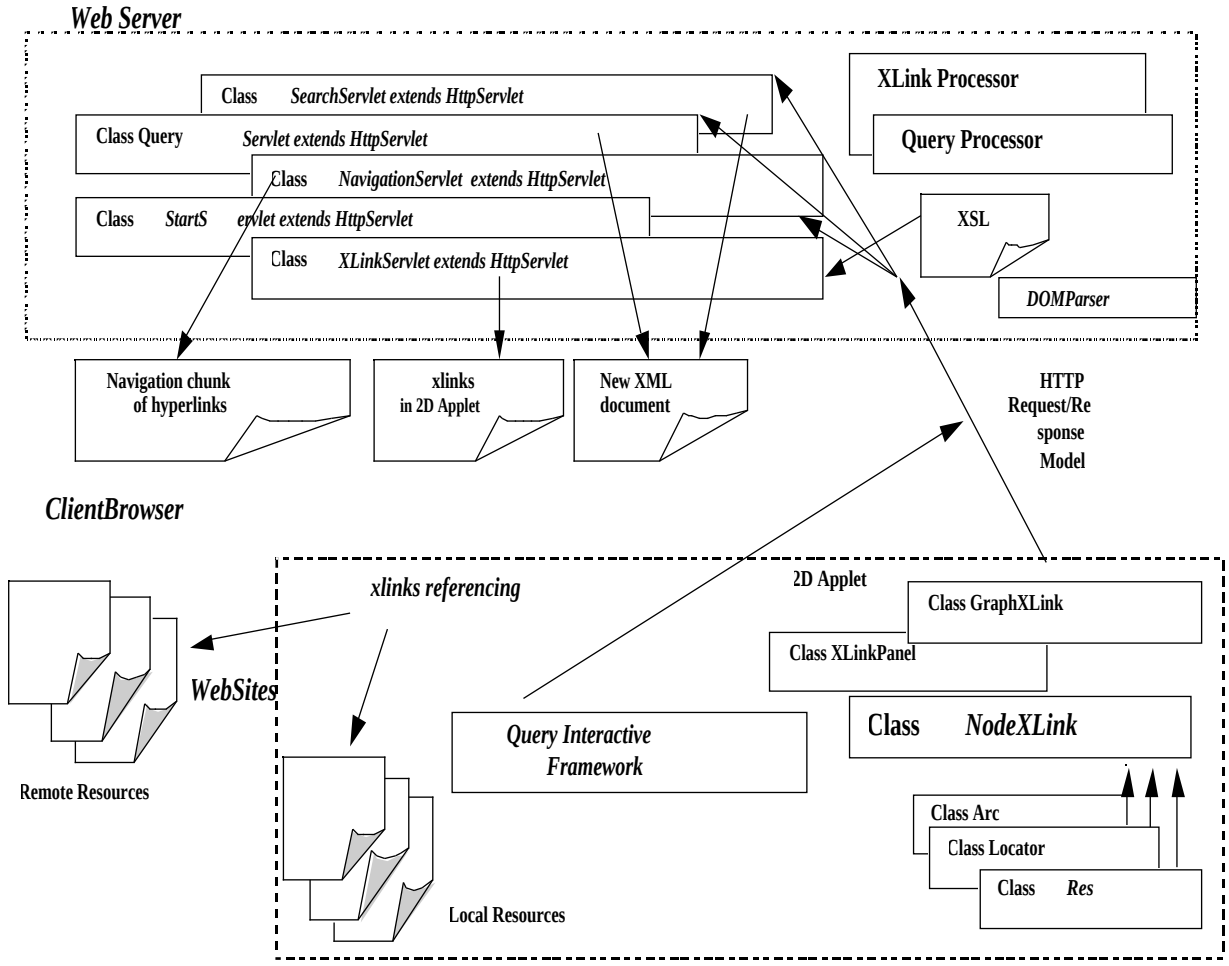

Fig.1. Java Client/Server framework for XLink traversing using DOM parser

Servlets are modules that extend request/response-oriented web servers as handling HTTP client requests for XLink parsing and processing data POST-ed over HTTP. The set of servlets also need to integrate a DOM Parser (in the present work the Xerces DOM Java Parser 1.4.1 is used):

```
import XMLParsing.*;
import org.w3c.dom.*;
import org.apache.xerces.dom.*;
import org.xml.sax.*;
public class XLinkServlet extends HttpServlet {
   public void doGet(HttpServletRequest request,
                HttpServletResponse response)
      throws IOException, ServletException
   {          String XML_Name = request.getParameter("XMLname");
              org.apache.xerces.parsers.DOMParser parser =    new org.apache.xerces.parsers.DOMParser();
           parser.parse(XML_Name);
      Document document = parser.getDocument();
```

The XLink servlet integrates an XLink processor and responds to the client browser through Java applets. It applies the XLink Processor to convert the XLink in XML document to parameters of xlinks applet.

```
…         //XLink Processor class is instantiated
          TraverseXLinks con1 = new TraverseXLinks(document);
          ResXML = con1.Return_ResXML();
          xlinks = con1.Return_Xlinks();
…
 response.setContentType("text/html");    PrintWriter out = response.getWriter(); out.println("<html>");    …
out.println("<applet          CODEBASE=http://starpc8.vub.ac.be:8080/XMLNav/Graph/          CODE=GraphXLink.class
width=900 height=220> ");            out.println("<param " + xlinks + ">");     out.println("</applet>");
```

An XLink processor reads the entire document and extract and store any extended links found there in a DOM tree. The knowledge for how to match the XLink information for synthesizing applet parameters is embedded in the XLink Processor:

- An extended link consists of a set of resources and a set of connections between them using the "arc" and "locator" attributes that define the path that one can take to reach a destination node;
- Attaching xlink:title attribute to the locator elements specifies Semantics;
- Extended links provide many different possible traversal paths. Each of these possible paths between resources can have different rules for when the link is traversed and what happens when it's traversed. Traversal rules are specified by attaching xlink: actuate and xlink: show attributes to arc elements. Applications can use arc elements to determine which traversals are and are not allowed and when a link is traversed;
- A single arc element may actually describe multiple arcs. If more than one resource has the xlink: label A, then arcs go from all resources with the label A to the resource with label B. Each arc goes from exactly one resource to exactly one other resource.

Appendix A shows Java classes of XLink and Query Processors.

The mapping of XLink to applet parameters in HTML tag results in the statements shown below:

<applet code="GraphXLink.class" width=350 height=200> <param name=Xlinks value="<with_extendedL> NAME(xlink:label)C_book,START_P(xlink:href)http://www.amazon.com/exec/,CUSTOMER_P(xlink:label)Customer PageCust,CUSTOMER_P(xlink:title)onCustomerPage,TARGET(xlink:href)http://images.amazon.com/images/P/15826. jpg,TARGET(xlink:label)info_targetCust,TARGET(xlink:title)has_title,…, CONNECTION(xlink:show)replace,CONNECTION(xlink:to)CustomerPageCust,CONNECTION(xlink:from)StartPage Cust, CONNECTION(xlink:to)info_targetCust,”> **</applet>**

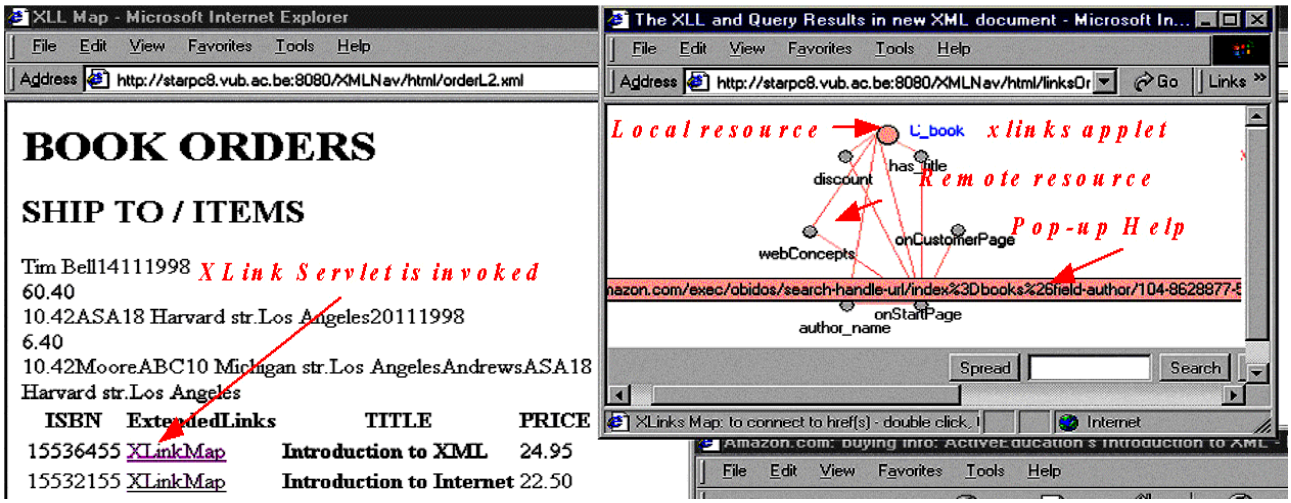

Fig.2. a. The transformed XML/XLink document b. the xlinks applet in a browser session

The xlinks applet converts these parameters to it's own object-oriented structure for XLink:

```
class NXLink {
   String type="extended";
    int numbLocator, numbAttr;
   Res res;
   Locator loc;
   Arc arc; }

class Res extends NXLink{
   String type="resource";
   double x, y;
   boolean selected;
   String role,title, label; }

class Locator extends NXLink{
   String type="locator";
   double x[]=new double[50];
   double y[]=new double[50];
   boolean selected[]=new boolean[50];
   String href[]= new String[50];
   String role[]= new String[50];
   String title[]= new String[50];
   String label[]= new String[50];}

class Arc extends NXLink{
   String type="arc";
   String from[]= new String[50];
   String to[]= new String[50];
   String show[]= new String[50];
   String actuate[]= new String[50];
   String arcrole[]= new String[50]; }
```

The end-user can click over the graphical notation of title attributes of the potential extended xlinks presented in the applet. If no direct arc exists between the local and remote resources the hyperlinks are associative: A link from *xlink:href1* to *xlink:href* 2 will only be traversed if you are at *xlink:href1*.When extended XLink is requested the browser sends that request to a Navigation Servlet, which responds to the client browser through HTML pages composed of two frames. At the top frame a chunk of hyperlinks for the requested navigation is presented. The hyperlinks are associated and creates more complex linking structure embedded in the extended XLink. For instance, the content of a tailored page may be different for different users based on the identity of the user or the value of one or more input fields. In dynamic pages the content may change depending on the navigation, e.g. a shopping basket page.

```
public class NavigationServlet extends HttpServlet {
            public void doGet(HttpServletRequest request,…
           String Query[]=new String[20]; //contains all  hyperlinks of entemded xlink
           FileWriter fw4 = new FileWriter("./XMLNav/top_frame.html");
  for (StringTokenizer t = new StringTokenizer(query, ",") ; t.hasMoreTokens() ; ) {
              String str = t.nextToken();          Query[i]=str;          i++;
           }
```

// Creating the first frame - the navigation chunk – file top_frame.html with Java script code for how to follow the hyperlinks

```
top_frame="<html> <head> <script language=JavaScript>" function preload(){ "
function showhide(what,what2)" + "\n" + "{if (what.style.display=='none'</script >  <BODY > <a
href=http://starpc8.vub.ac.be:8080/XMLNav/html/All_XLL.html target=out>XLL</a> <img id=menu1sign
src=back.gif valign=bottom> <a href=javascript:history.go(-1);>… style="
+ "\""+"cursor:hand; font-Family:Verdana; text-decoration:underline; font-weight:bold"
+ "\""+"><font style=text-decoration:none><img id=menu1sign src=index.gif valign=bottom> </font>Navigation
Chunk</span><br>"+ "<span id=menu1outline style=display:'none'> ";
```

// Creating the chunk of associate hyperlinks

```
for (int j=i-1; j>0; j=j-2) {
top_frame = top_frame+ " <a href= "+ Query[j-1] + " target=out>" + Query[j]
 + "</a> <img id=menu1sign src=transfer.gif valign=bottom> ";
 }
```

// Dynamic HTML with two frames – top_frame.html is appear in the target "in", the requested resource is placed in the target "out"

```
response.setContentType("text/html");    PrintWriter out = response.getWriter();     out.println("<html>");
out.println("<head>");
out.println("<FRAMESET rows=14,80> <FRAME src=http://starpc8.vub.ac.be:8080/XMLNav/top_frame.html
scrolling='no' frameborder='0' name=in>");
out.println("                <FRAME src=" + Query[i-2]+ " frameborder='0' name=out>");
out.println("    </FRAMESET>"); out.println("</html>");
   }
```

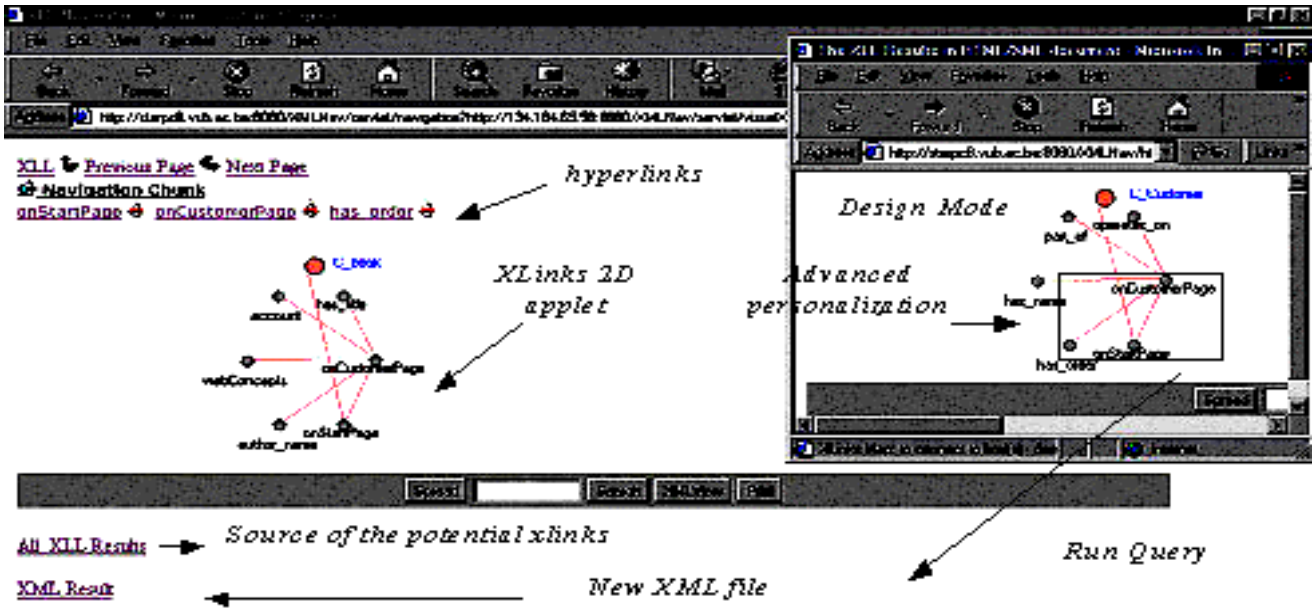

Fig.3. The Query Interactive Framework (Design mode)

Advanced personalization of the XLink document can be performed on the client side. The main function of the visual presentation, in the applet, is to perform graphical XLink querying, not just to improve the user's point of viewing. Since the titles of the locators define the semantics and are more valuable than the information they link, the process of xlink:title definition in the XML document is very important. The idea implemented in the Object Role Model [3], that the world is comprised of objects playing roles, is used here in a new way: part of the xlink:titles are verbs corresponding to the roles the connected resources play in relation to each other. One can choose among several alternative verbs to prepare statements in a way very similar to the natural language expression. The querying is performed through dragging, clipping and matching the graphical

notation of the extended xlink:titles*(Fig.3.)*. Performing the interactive process of querying is straightforward by using the Java 2D Graphics API, i.e.:

```
paintLink(Graphics g, NXLink n, FontMetrics fm){     g.drawOval (n.loc.x,n.loc.y,…);
mouseClicked(MouseEvent e){         rectangle.contains(n.res.x,n.res.y);..}

GraphPanel panel;     Panel queryPanel;     Panel controlPanel;
queryPanel.add(New_Window); New_Window.addActionListener(this);
queryPanel.add(run_query); run_query.addActionListener(this);
```

Editing the values of the xlink:attributes in the control panel of the applet modifies the behavior of the xlinks. One usually chooses multiple objects and verbs. One can drag, clip and match multiple statements to prepare a complex query for navigation that corresponds to specific tasks performed within a specific hypermedia system. The *QueryServlet* is invoked during the actions in design mode. It forwards the new combination of XLinks to the Query Processor. The result of the query composed this way is a new well-formed XML document(s) with new information for XLink elements. After the XML file is prepared, the user can observe the effect of a new edition of XML document. In the design mode the user can:

- Explore the different guided tours made possible by the xlinks and select the better ones;
- Change the arcrole, show and actuate value attributes corresponding to the number of opened windows for task completion;
- Check whether the concepts are reachable and if the target XML page is fully navigable;
- Analyze and reduce the longest path in hyperspace that one can take to reach the destination node.

4. Example

Let's consider a company ordering books online from Amazon.com. The department ordering books makes the expenditure, receives invoices and should be able to share part of the information with other departments. The tasks performed by the accountant after buying books on the Web are different from those of the librarian. The accountant's navigation chunk should involve web tasks related to Account Maintenance on "Amazon.com" like orders and transactions, trusted partners, option settings as well as local tasks related to read or announce information on company white-board; to reference the company databases or documents and regulations *(Fig.4.)*. On the other hand, the librarians should be informed of the acquisition and have to enter the related data into the database of the company. They deals with information pertaining to the Amazon.com line of products such as title, ISBN number, name of the author, users' feedback and opinions etc.*(Fig.5.)*. The book ordering is a recurrent process iterating a different set of URIs for each task completion. Extended links remove the requirement to insert link information inside the document content. All potential URIs should be part of one XML document. The advantage of that XML link base is that every time the referencing documents are moved, renamed or divided into smaller pieces, the links only need to be adjusted in one XML file. It should integrate the local and remote resource for referencing the information or services for e-paying; e-forms filling; database updating; white-board writing, etc. For the time being it would be managed manually by the Information department of the company. Also intelligent agents should be able to compose the XML files. Intelligent agent needs only to update the metadata for XLink elements by harvesting the URLs and finding out the titles of the xlinks. The result xlink applet with all potential local and remote resources is shown (*Fig.2*.). The possible personalization can be performed in the Query Interactive Framework (*Fig.3*.). After that each department obtains it's own XML document(s). The navigation obtained in this way leads to a navigation chunk of associate hyperlinks for task completion according to the specific needs of

different departments. A personalized XML document for the accounting department comprises the accounting information with local and remote extended xlinks. The locators will take one to the "Account Maintenance" of Amazon's store as well as to company's database or documents and regulations *(Fig.4)*. An XML document comprises the book's information with extended xlinks for the librarian. The locators of the xlinks will take one to the main sections of Amazon's bookstore as well as to the company's private book database or documents and regulations *(Fig.5.)*.

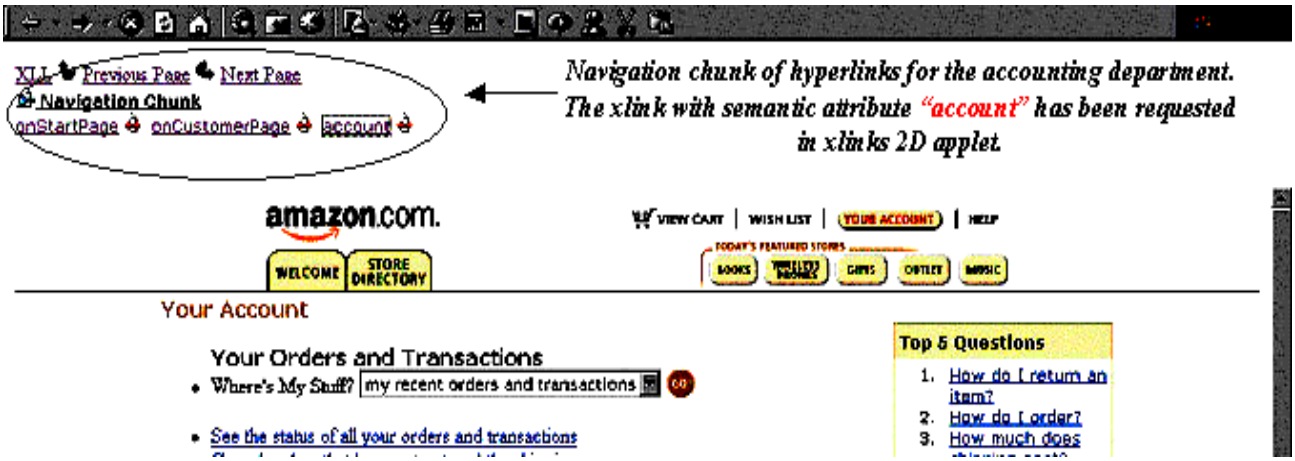

Fig.4. Semantic Navigation chunk for the accounting department

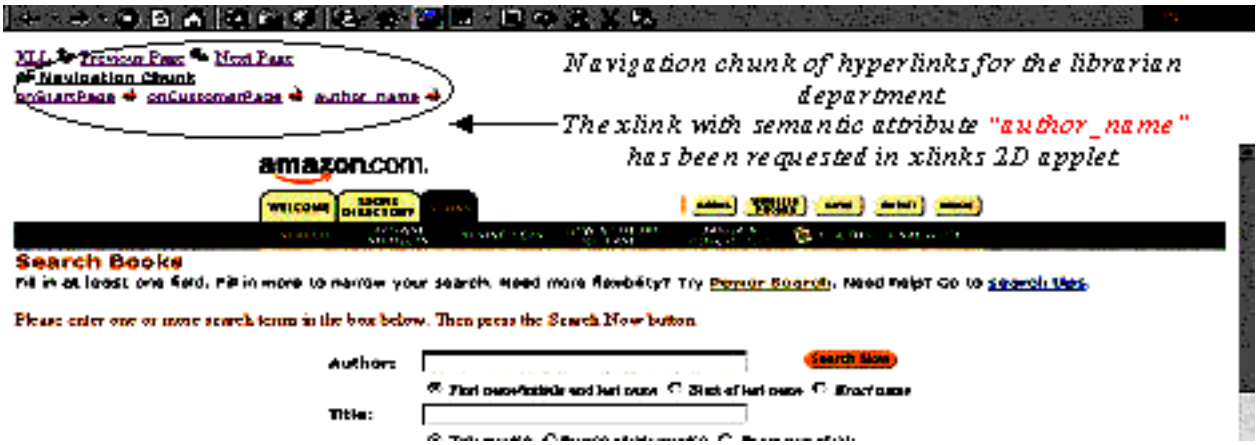

Fig.5. Semantic Navigation chunk for the librarian department

5. Conclusion

The proposed approach creates the effect of having a dynamic, virtual temporary XML browser for semantic navigation and organization of the information over the Internet. It will improve the web-based online services while referencing the existing electronic documents. A browser generates web GUIs that is easier to use, both from the designer's and the user's point of view, as it employs interactive web tools for traversing and xlinks managing in graphical mode presentation. To do so, the HTTP server traverses the set of xlinks for the HTTP client, which set dynamically a semantic navigation chunk of hyperlinks in the manner in which specific tasks are performed.

References

[1] BLOCH, S., Bodoff, S., Servlets, Java Tutorial, http://java.sun.com/docs/books/tutorial/servlets/index.html

[2] FUJITSU, XLink Processor, http://www.fujitsu.co.jp/hypertext/free/xlp/en/
[3] HALPIN, T. (1995), *Conceptual Schema and Relational Database Design*, Second Edition Prentice
[4] HAROLD, E R., The XML Bible, XLinks, http://www.ibiblio.org/xml/books/bible/updates/16.html
[5] JUSTIN, L., *Link*an XML–XSL-XLL browser, http://pages.wooster.edu/ludwigj/xml/index.html
[6] Tools for processing XLinks and Xpointers, http://www.xmlsoftware.com/xlink/
[7] W3C Candidate Recommendation - Extensible Stylesheet Language, http://www.w3.org/TR/xsl/
[8] W3C Candidate Recommendation - XML Linking Language Xlink, World Wide Web Consortium, 2000, http://www.w3.org/TR/xlink.)
[9] X2X - XLink engine, http://www.stepuk.com/products/prod_X2X.asp
[10] XLink Filter Project**,** http://www.simonstl.com/projects/xlinkfilter/

Appendix A: Java application for XML parsing