Information Technology and Management 3, 271–290, 2002

Using XML for Supplemental Hypertext Support

CHAO-MIN CHIU cmchiu@ccms.nkfu.edu.tw
Department of Information Management, National Kaohsiung First University of Science and Technology, 1 University Rd. Yenchao, Kaohsiung, Taiwan

MICHAEL BIEBER bieber@njit.edu
Collaborative Hypermedia Research Laboratory, Computer and Information Science Department, New Jersey Institute of Technology Newark, NJ 07102, USA

Abstract. Our overall research goal is providing hypertext functionality through the WWW to hypertext-unaware information systems with minimal or no changes to the information systems. Information systems dynamically generate their contents and thus require some *mapping mechanism* to automatically map the generated contents to hypertext constructs (nodes, links, and link markers) instead of hypertext links being hard-coded over static contents. No systematic approach exists, however, for building mapping routines to create useful links that give users direct access to the ISs' primary functionality, give access to metainformation about IS objects, and enable annotation and ad hoc (user-declared) linking. This paper contributes a procedure for analyzing ISs and building mapping routines that supplement information systems with hypertext support. This paper also contributes an eXtensible Markup Language (XML) DTD that declares a set of elements and attributes for representing mapped information in a human-readable, machine-readable, structured, and semantic way. We implemented a prototype to demonstrate the feasibility of using XML to represent mapped information.

Keywords: hypertext, information systems, mapping mechanism, eXtensible Markup Language, Wrapper, Relationship-Navigation-Rule analysis, World Wide Web

1. Introduction

Our overall research goal is providing hypertext functionality through the World Wide Web (WWW) to hypertext-unaware information systems (IS) with minimal or no changes to the ISs. ISs include financial information systems, accounting information systems, expert systems, decision support systems, etc. ISs dynamically generate their contents and thus require some *mapping mechanism* to automatically map the generated contents to hypertext constructs (nodes – documents and screens, links – commands and relationships, and link markers – for selecting links) instead of hypertext links being hard-coded over static contents [14].

What benefit do users gain from providing information systems with hypertext support? Users may find it difficult to understand and take advantage of the myriad of inter-relationships in an information system's knowledge base (data, processes, calculated results, and reports). Hypertext helps by streamlining access to, and providing rich navigational features around related information, thereby increasing user comprehension

of information and its context [6]. Augmenting an information system with hypertext support results in new ways to view and manage the information system's knowledge, by navigating among items of interest and annotating with comments and relationships (links) [7,8].

No systematic approach exists, however, for building mapping routines to integrate an information system with the WWW and give users direct access to interrelationships among objects of the information system. This paper proposes a systematic dynamic-mapping mechanism for mapping outputs from an information system to hypertext constructs.

Why do we use XML for representing mapped information? There are three major text-based markup languages for interchanging and publishing documents in a standard and open format: Standard Generalized Markup Language (SGML), eXtensible Markup Language (XML), and HTML. SGML and XML are well-accepted and standard metalanguages for describing markup languages. SGML allows the development of custom and domain-specific document formats through Document Type Definitions (DTDs). A DTD is a set of rules defining the logical sequence of elements within an SGML or XML document and specifying the allowed content and attributes of each element. However, SGML is too complicated for most WWW use due to its overwhelming number of options and customization features [12,19]. XML [11] is a subset metalanguage of SGML designed specifically to deliver information over the WWW [19,23]. HTML is a markup language and is therefore called an "SGML application", as opposed to a metalanguage. HTML is for marking up document elements, but not for semantic metadata. (Metadata provides information about the data and the elements within a document, helpful in conveying the meaning and attributes about each in a machine-readable format.) HTML only supports a fixed and limited tagset that conveys display information about elements, such as making them boldface. ISs need to process data, not only display them. Therefore, when reengineering IS applications for the Web, XML is recognized as a better approach than HTML because it offers extensible, human-readable, machine-readable, semantic, structural, and custom markup.

Resource Description Framework (RDF) is an application of XML which goal is providing interoperable and machine-understandable metadata about Web resources (e.g., Web pages) [24]. RDF enables the consistent encoding, exchange, and reuse of structured metadata and automated processing of Web resources [24]. We define our own XML DTD instead of using RDF for two reasons. First, RDF provides a general syntax for expressing metadata. RDF itself does not provide any predefined vocabularies for authoring metadata [10], which our XML DTD does. However, some of vocabularies (e.g., "Dublin Core") will emerge in the foreseeable future. Second, browsers that support RDF were not available when we implemented the prototype.

This paper makes two contributions. First, we propose a procedure for analyzing ISs and building mapping routines. Mapping routines create useful links that give users direct access to the ISs' primary functionality, give access to metadata about IS objects, and enable annotation and ad hoc links. Second, we define a XML DTD that declares a set of elements and attributes for representing mapped information generated by in-

formation systems. We have implemented a prototype to demonstrate the feasibility of using XML to represent mapped information.

This paper is organized as follows. Section 2 reviews some approaches to integrating information systems with the WWW. In section 3 we propose a conceptual architecture for interfacing ISs with the Web. Section 4 presents a procedure for building mapping routines and specifies our XML DTD. In section 5 we discuss future work. Section 6 concludes with a larger view of our research.

2. Integrating information systems with hypertext support

Hypertext researchers have developed several hypertext features to help users easily navigate information, and reduce cognitive overhead and disorientation (i.e., becoming "lost in the hyperspace" [18]). Those supports include backtracking [27], history lists [29], guided tours [28], overview diagrams [17,29], paths [28], structure-based query [25], timestamps [26], footprints [26], fisheye viewer [20], annotation [2,13], etc.

Many approaches exist for integrating information systems into the WWW, such as Common Gateway Interface (CGI) and Active Server Pages (ASP) used to interface the WWW server and external programs and scripts. Both CGI scripts and ASP programs are invoked by a WWW server to process users' inputs and generate HTML documents dynamically. No systematic approach exists, however, for dynamically supplementing an information system with hypertext support through the WWW and giving users direct access to its interrelationships. Many information systems resources have been made available to users through the WWW. However, those implementations do not meet our criteria for integration with ISs since they do not create useful links that give users direct access to the ISs' primary functionality, give access to metainformation about IS objects, and enable annotation and ad hoc links. For example, the following cases do not meet our criteria:

- Most Web database applications handle queries and generate HTML documents from query results without mapping query results to useful links that allow users to view and manage the DBMSs' contents, navigate among related items of interest or annotate with comments and links.
- WWW search engines use simple mapping mechanisms to map a query result to a dynamically generated Web page with links based on the "URL address" and "page title" for each hit.
- The Web Interface Definition Language [3] is an application of the XML that provides interfaces and services to automate the process of information access to remote applications and systems. Web Interface Definition Language services are like CGI scripts or other back-end Web server programs. A service takes input parameters, performs some processing, and then returns a dynamically generated HTML, XML, or text document to Web browser for display.

Only two systematic approaches exist for reengineering applications for the WWW: Bieber's two-stage Web Engineering [5] and our four-step RNRA approach.

With Web Engineering, first the software engineer performs a *Relationship-Navigation Analysis* (RNA), analyzing the application specifically in terms of its intra- and inter-relationships. RNA has 5 steps: stakeholder analysis, element analysis, relationship and metainformation analysis, navigation analysis, and relationship and feasibility analysis. Second, a Dynamic Hypermedia Engine (DHymE) automatically generates links for each of these relationships and metainformation items at run-time, as well as sophisticated hypermedia navigation techniques not often found on the Web (e.g., guided tours, overviews, and structural query [9]) on top of these links.

The major difference between Web Engineering and our RNRA approach is that we clearly define the transition from determining the relationships to implementing the mapping. RNA supplements our mapping mechanism. Combining RNA and our mapping routine approach forms our *Relationship-Navigation-Rule Analysis* (RNRA) technique for engineering applications for the World Wide Web.

We have successfully implemented a prototype to demonstrate the feasibility of using our RNRA approach to analyze and XML to represent mapped information. Bieber currently is integrating several applications with DHymE, automatically giving each a Web interface or supplementing its existing Web interface. A major difference between our prototype and Bieber's DhymE is the way of displaying mapped information (i.e., XML documents). Our prototype uses eXtensible Stylesheet Language (XSL) to display XML documents. However, the DhymE's browser wrapper converts XML documents to HTML format for display. The contribution of this paper is on the RNRA approach. RNRA easily would support integrating applications with Bieber's engine as well as any other systems that would map hypermedia functionality dynamically to applications. But in fact, no other general approaches or prototypes have ever been developed to reengineer applications for the WWW by providing run-time mapping of application information and relationships to hypermedia constructs.

3. System architecture

In this section, we propose a framework with seven logical components. This architecture emphasizes the integration of the WWW with ISs, providing hypertext functionality to each IS. Figure 1 sketches the architecture with three example ISs. Different implementations may implement these logical components in different ways. Our prototype demonstrates one possible implementation. Since the WWW browser and server are normal WWW components, we will only describe the functionality of other components:

- An IS is an application system, with which users interact to perform certain tasks. As a result the IS dynamically produces output content for display. ISs are often application packages. IS instances are written within an IS package (e.g., individual worksheets are instances of a spreadsheet, and a database in an instance of a database management system).
- An IS wrapper translates and routes messages between its IS and the WWW server. An IS wrapper also provides information to map links and nodes on top of the out-

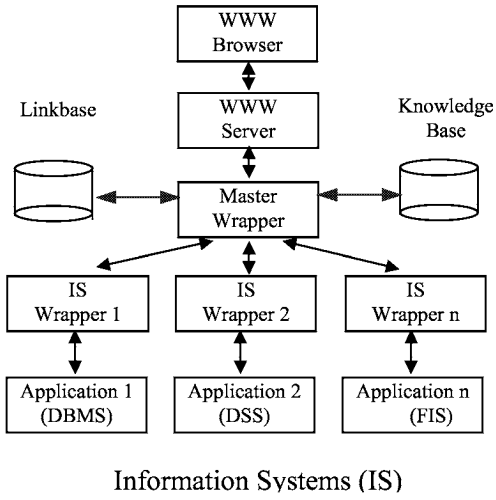

Figure 1. A framework for integrating information systems into the WWW.

puts of its IS. An IS wrapper must map commands selected by the user on the WWW browser to the corresponding IS commands and invoke its IS to execute them. A comprehensive IS wrapper will allow us to integrate an existing IS with few or no changes.

- The knowledge base stores commands for accessing various relationships on IS objects that cannot be accessed directly from ISs (e.g., relationships in entity–relationship diagrams).
- The linkbase stores user-created annotations and ad hoc links.
- The master wrapper coordinates relationship mapping and message passing among different IS application domains, thus aiding IS-to-IS integration. It provides the following functionality: (1) decodes attributes (e.g., object type, object ID and command) that underlie a link anchor, (2) searches the knowledge base for commands that implement various relationships from the selected IS object based on the object type, (3) maps commands to link anchors, and (4) forms an XML document that includes the mapped link anchors and send the document to the WWW server. Note that the master wrapper can be implemented with CGI programs, server-side Java, ASP, etc.

To integrate a new IS with the WWW and provide it with hypertext support, one has to build a wrapper, declare mapping routines within it and the master wrapper, and store information (e.g., commands and access information) in the knowledge base. One set of mapping routines can serve all instances of an IS.

We use two mapping routines: Command_Routine and Object_Routine. How do mapping routines work? Suppose that within our financial analysis system shown in figure 5, a user wants to conduct the Earnings Before Income Tax–Earnings Per Share (EBIT-EPS) analysis for 1998. First the user clicks on the hyperlink labeled "Y1998." The Object_Routine will be invoked by the Web server to search for available commands, map them to hyperlinks, and then deliver an XML document containing them

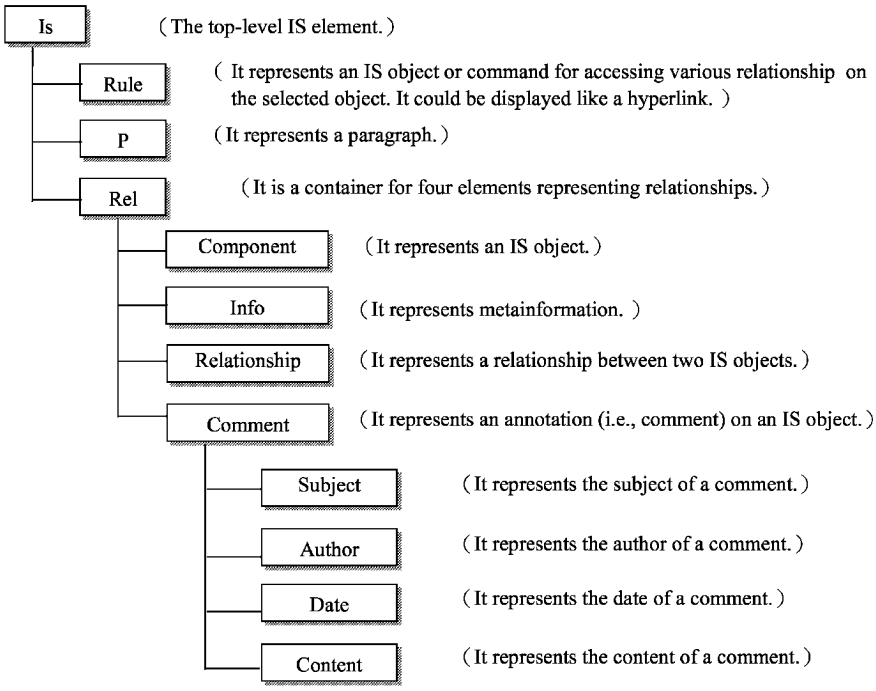

Figure 2. The logical structure of our DTD.

to the browser for display. Then the user clicks on the command hyperlink labeled "ShowData." The Command_Routine will be invoked to execute the "ShowData" command. The Command_Routine will send parameters to the financial information system, receive results of the EBIT-EPS analysis, mark up information objects within it as hyperlinks, and then send the resulting XML document to the browser for display.

4. Building mapping routines

In this section, we present a four-step process for analyzing ISs and building mapping routines that convert dynamically generated information to hypertext constructs (nodes, links, and anchors). These routines reside in and become invoked by the master wrapper or IS wrapper. We explain each by using our Financial Information System in Microsoft Excel (FISME) as the target IS. When providing large information systems with hypertext support, a facility that automatically creates useful links from dynamically generated information will be helpful. Mapping routines make extensive use of three features that Halasz [22] identified among outstanding issues in hypermedia research over ten years ago, and which still have not been addressed in many hypermedia or WWW applications: (1) creating and manipulating virtual structures of hypermedia components; (2) computing over the knowledge base during link traversal; and (3) tailoring the hypermedia network [4]. Information systems in which component type or classes are easily

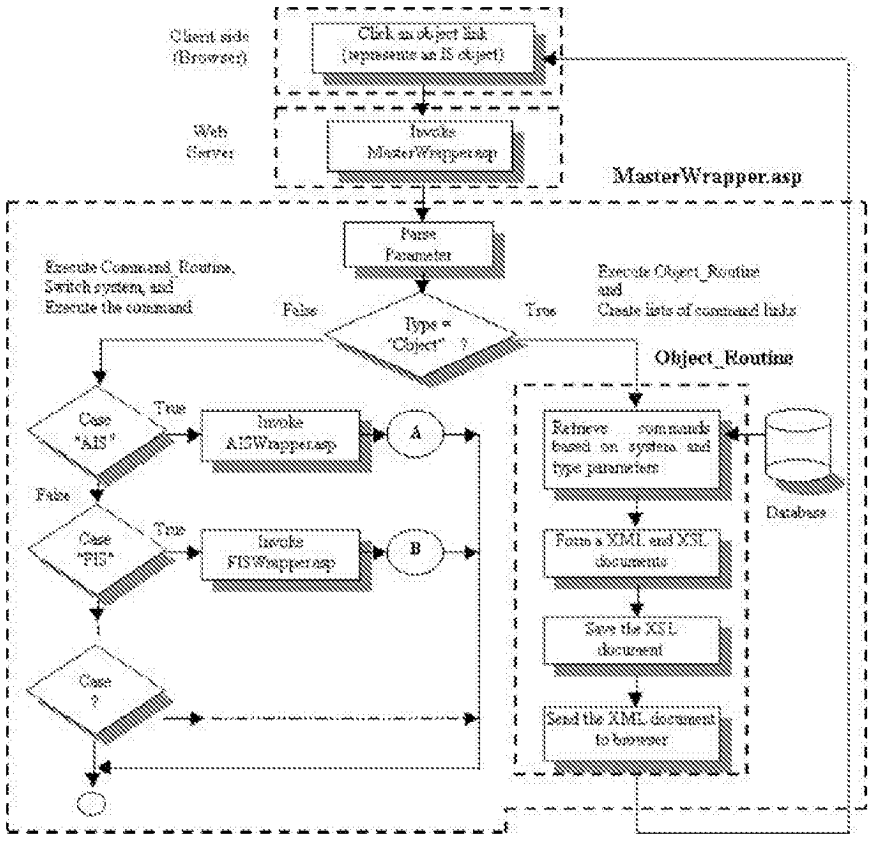

Figure 3. The program flow of MasterWrapper.asp. Note that AIS stands for accounting information systems and FIS stands for financial information systems.

recognized can benefit most easily from the automatic link generation approach [21]. Many of these systems have application programming interfaces (APIs), which facilitate integration with a wrapper.

4.1. Step 1. Identify IS objects

This step is to identify data objects in which we are interested. In our Financial Information System in Microsoft Excel (FISME), objects include workbooks, worksheets, and cells. This corresponds to RNA's element analysis stage, which currently awaits development [30,31].

4.2. Step 2. Identify relationships among IS objects

Bieber and Vitali [8] and Yoo and Bieber [30,31] identify several types of relationships for system objects. Identifying these explicit and implicit relationships forces developers to consider which information users are interested in and then build mapping routines to

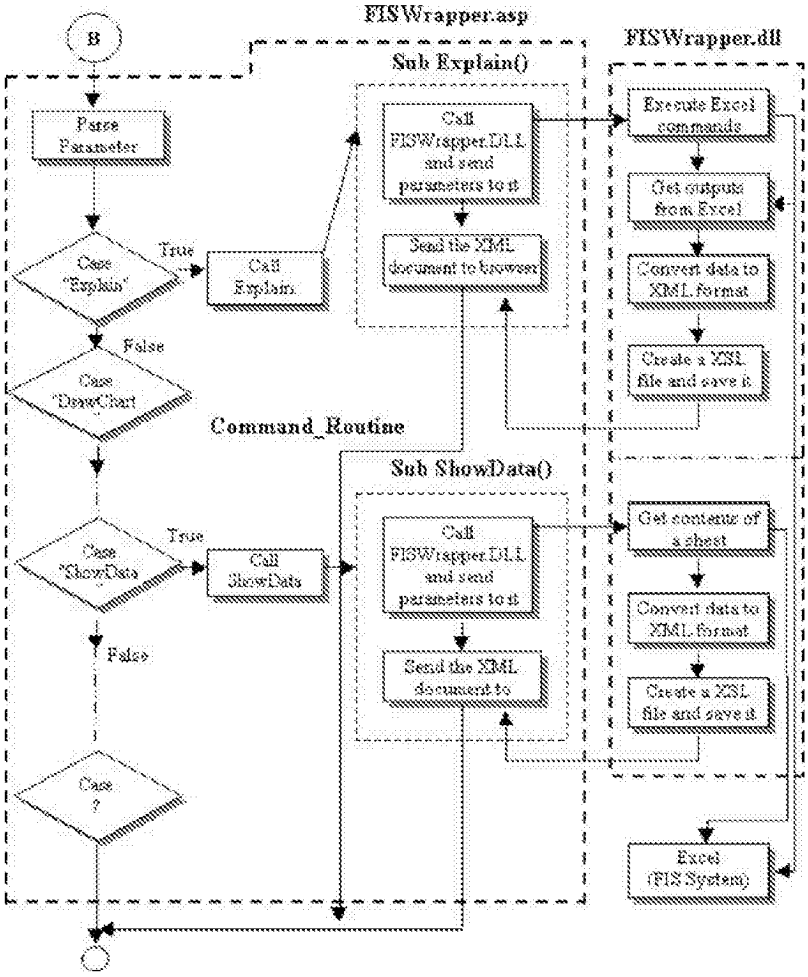

Figure 4. The program flow of FIS Wrapper (i.e., FISWrapper.asp and FISWrapper.dll combined), detailing the "Explain" and "ShowData" commands.

access this information. This corresponds to the relationship analysis stage of RNA. Yoo and Bieber [32] and Yoo [30] provide detailed guidelines for conducting a relationship analysis.

Each of the following relationships gives the user easy access to some aspect of an object. Sometimes the destination of meaningful relationships are not found within any IS, so developers have to declare the methods to execute them and store these in the knowledge base. We have implemented the following five relationship types explicitly in our prototype, as a proof of concept:

- *Schema relationships*: Access to the kind of domain-specific relationships in a schema or application design.
- *Operational relationships*: Direct access to IS objects resulting from or arrived at

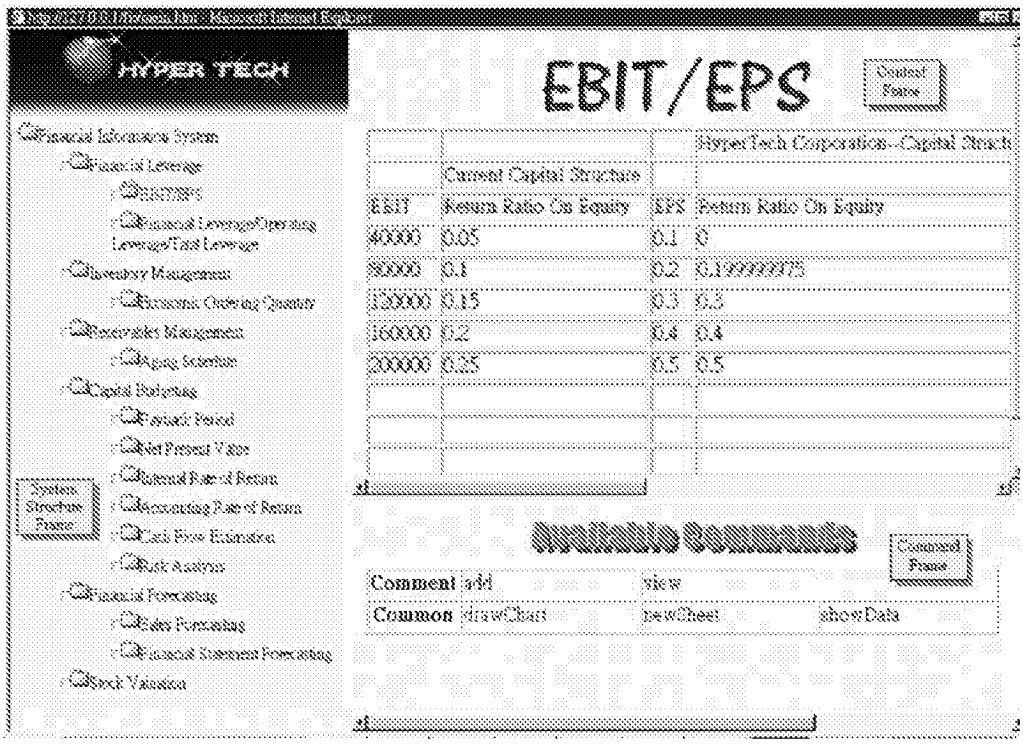

Figure 5. This figure shows the interface of our prototype. This system has three frames: system structure, command, and content.

from using operational commands supported by the IS. In our FISME prototype, this includes Excel commands over specific objects.

- *Structural relationships*: Access to related objects based on the application's internal structure. In our FISME prototype, these include "contains" links among workbooks, worksheets, and cells.
- *Metainformation relationships*: Access to attributes of and descriptive information about IS objects. In our FISME prototype, these include its formulas, explanations, cell types, etc.
- Annotative relationships: Relationships declared by users instead of being mapped from the system structure. All users should be able to annotate objects even when the application does not allow direct write access. A linkbase will store user-created links and comments.

We have defined an XML DTD that declares a set of elements and attributes for representing relationships. Appendix A contains the full DTD along with an explanation of each keyword. Figure 2's XML element tree shows the logical structure of our DTD.

The three examples in tables 1–3 demonstrate how to use our elements and attributes to represent various relationships. Note that FIS stands for financial infor-

Table 1
An XML document representing structural relationships in our FISME prototype. These represent some components of the system structure frame in figure 5.

```
<? XML VERSION = "1.0" ?>
<! DOCTYPE Is SYSTEM "Is.DTD">
<Is>
<Rel Type="Structural" ObjectID="FIS">
   <Component Id="FIS,FL" Name="FL" Level="1">Financial Leverage</Component>
   <Component Id="FIS,FL,EBIT-EPS" Name="EBIT-EPS" Level="2">EBIT/EPS</Component>
   <Component ID="FIS,FL,FL-OL-TL" Name="FL-OL-TL" Level="2">Financial
    Leverage/Operating Leverage/Total Leverage</Component>
. . . . .
</Rel>
</Is>
```

Table 2
An XML document representing metainformation relationships for a FISME cell. These represent components of the pop-up window in figure 6.

```
<? XML VERSION = "1.0" ?>
<! DOCTYPE Is SYSTEM "Is.DTD">
<Is>
<Rel Type="Metainformation" ObjectID=" FIS,FL,EBIT-EPS,Y1998,C3">
   <Info Type="Text">EPS: A company's net profit minus its preferred stock obligations, with the
    difference divided by the number of outstanding shares of common stock.</Info>
</Rel>
</Is>
```

Table 3
An XML document that representing an annotative relationship for a FISME worksheet. These represent components of the comment in figure 10.

```
<? XML VERSION = "1.0" ?>
<! DOCTYPE Is SYSTEM "Is.DTD">
<Is><Rel Type="Annotative" ObjectID="FIS, FL,EBIT-EPS,Y1998">
      <Comment>
        <Subject>More Explanation</Subject> <Id>FIS, FL,EBIT-EPS,Y1998</Id>
        <Author> C.M. Chiu</Author> <Date>4/20/1999 02:17:36 PM</Date>
        <Content>Trends in EPS can be used to identify potential earning dilution on a per-share
          basis.</Content>
      </Comment>
  </Rel></Is>
```

mation system. The tables describe XML documents that underlie the system structure frame of figure 5, the pop-up window in figure 6, and figure 10, respectively. Figure 5 shows the interface of our prototype. The interface contains three frames: system structure, command, and content. The system structure frame lists all finan-

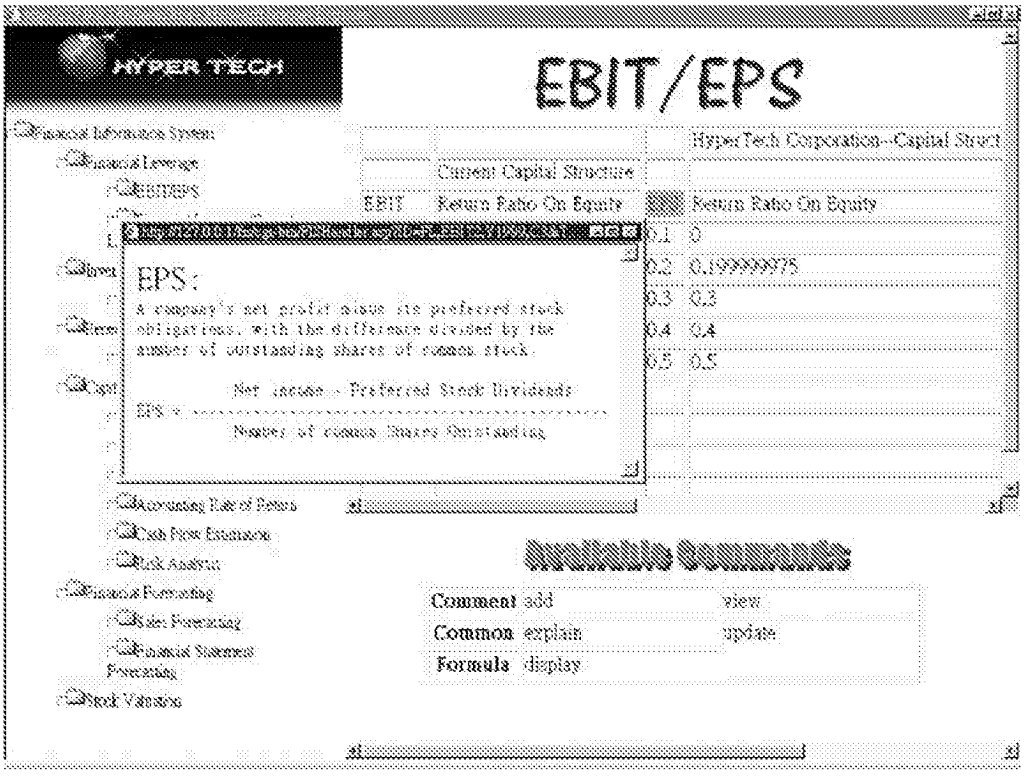

Figure 6. Output of executing the "explain" command.

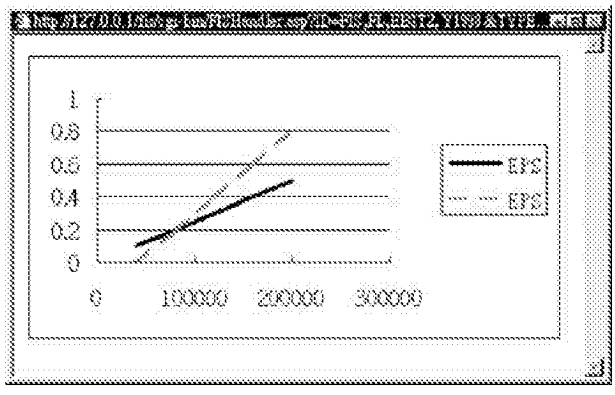

Figure 7. Output of executing the "DrawChart" command.

cial analysis functions of our prototype, which include EBIT/EPS, financial leverage, operational leverage, etc. figures 6–10 show outputs of executing some system commands.

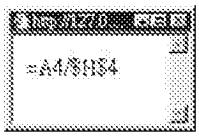

Figure 8. Output of executing the "display formula" command.

Figure 9. A form for users to add comments.

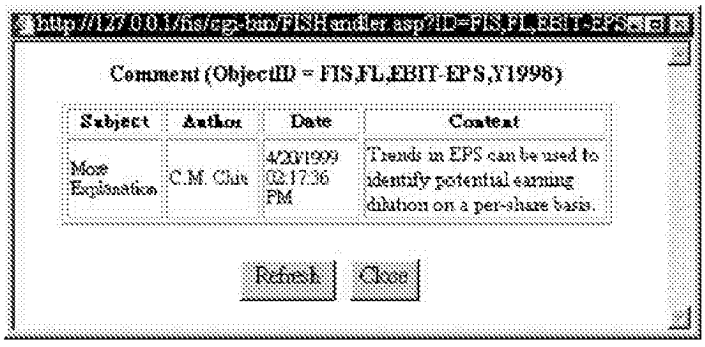

Figure 10. Output of executing the "view comment" command.

4.3. *Step 3. Identify commands for accessing each relationship*

The commands underlying the <Rule> tags (see table 5) give users direct access to various relationships on IS objects. In this step we need to identify the actual implementation commands for every relationship found in step 2. Then we can build the relationship mapping routines. Table 4 lists some of these commands for worksheet and cell relationships. The display labels for these commands may be different from the actual system commands that the IS wrapper passes to the IS for execution.

Table 4
Selected commands underlying worksheet and cell relationships.

Object	Relationships	Commands	Functionality
Sheet	Operational	DrawChart	Draw a chart based on values of the selected sheet
Sheet	Structural	ShowData	Show contents of a sheet
Sheet	Annotative	Add	Create comments on a sheet
Sheet	Annotative	View	View comments on a sheet
Sheet	Operational	NewSheet	Create a new Excel sheet
Cell	Operational	Update	Change the value of a cell and propagate changes
Cell	Metainformation	Explain	Get the explanation or formula of a cell
Cell	Annotative	Add	Create comments on the selected cell
Cell	Annotative	View	View comments on the selected cell
Cell	Metainformation	Display	Display the formula (in Excel format) of a cell

Table 5
This table lists commands for a FIS worksheet. Commands underlying the <Rule> tags give users direct access to various relationships on IS objects.

```
<? XML VERSION = "1.0" ?>
<! DOCTYPE Is SYSTEM "Is.DTD">
<Is>
Available Commands
<Rule Type="Command" Show="New" Href=http://127.0.0.1/fis/cgi-bin/MasterWrapper.asp
  ObjectID="FIS,FL,EBIT-EPS,Y1998" ObjectType="Sheet" Command="ShowData">
showData</Rule>
. . . .
</Is>
```

4.4. *Step 4. Build mapping routines*

We identify two mapping routines: Object_Routine and Command_Routine. Object_Routine is invoked to generate a list of links when the user selects an anchor with a <Rule> tag representing an IS object. Command_Rroutine is invoked to generate the link destination document when the user selects a command link. For clarity we discuss mapping routines in terms of functional procedural calls.

Figures 3 and 4 show the program flow of Object_Routine and Command_Routine. We use our FISME prototype as the example IS. The master wrapper is an ASP program (i.e., MasterWrapper.asp). The IS wrapper has two parts: an ASP program (i.e., FISWrapper.asp) and a DLL (i.e., FISWrapper.DLL) built using Visual Basic. ASP is Microsoft's browser-independent and server-dependent scripting language for interacting with external resources. An ASP program is a text file with an .asp extension that contains combination of XML statements and server-side scripts. Server-side scripts are written in VBScript, JScript, or other compliant scripting languages. Web servers that support ASP include Microsoft's Internet Information Server (IIS) and Personal Web Server (PSW). An ASP program is invoked by the Web server to process a user's re-

quest, dynamically generate XML statements, and then deliver them to the browser for display.

eXtensible Stylesheet Language (XSL) is currently being developed by the W3C XSL Working Group. XSL is a language for specifying the presentation of XML documents [1]. XSL consists of two techniques: transforming XML documents into another structure and specifying formatting semantics of the transformed information [1]. Internet Explorer 5.0 supports a subset of W3C's XSL (second XSL Working Draft). Our prototype uses the XSL technology of IE 5.0 to display XML documents and manipulate links.

4.5. Object_Routine(System, ObjectType)

This rule searches the knowledge base for commands accessing various relationships on the selected IS object and converts them to <Rule> tags (i.e., hyperlinks). This rule should be included in the master wrapper (see figure 3). As figure 1 shows, a WWW server can integrate with multiple ISs so we need the "System" parameter to discriminate among different ISs. The first parameter of ObjectID is the "System" parameter.

Object_Routine should provide the following functions:

(1) search the knowledge base for commands accessing various relationships on the selected IS object;

(2) map commands to <Rule> tags;

(3) form a XML document that includes mapped <Rule> tags and sends the document to the Web server.

For example, Object_Routine("FIS", "Sheet") will execute the aforementioned functions and create the XML document in table 5. As an example, we list the "ShowData" command here. Of course, the system would present all commands (or a filtered subset). Table 4 lists some available commands.

Note that the master wrapper is an Active Server Page (ASP) application called "MasterWrapper.asp." The <Rule> tag is displayed as a hyperlink. This type of <Rule> tag is called a command <Rule> tag since the value of the "Type" attribute is "Command." In the command <Rule> tag, the value of the "Command" attribute is "ShowData" instead of "No." When a user clicks on this link, the "FISWrapper.asp" application will be invoked and then the Command_Routine will be called.

4.6. Command_Routine(Command, ObjectID, ObjectType)

Command_Routine sends system commands to ISs and maps information objects dynamically generated by ISs to <Rule> tags (i.e., hyperlinks). This routine should be included in the IS wrapper (see figure 4). The "Command" parameter passes descriptive information about the actual IS command.

Table 6
Possible identifiers for FIS objects.

Object type	ObjectID structure	ObjectID example
Subsystem	System,Subsystem	"FIS,FL"
Workbook	System,Subsystem,Workbook	"FIS,FL,EBIT-EPS"
Sheet	System,Subsystem,Workbook,Sheet	"FIS,FL,EBIT-EPS,Y1998"
Cell	System,Subsystem,Workbook,Sheet,Cell	"FIS,FL,EBIT-EPS,Y1998,A3"

Table 7
This table lists <Rule> tags that represent objects of a FIS worksheet.

```
<? XML VERSION = "1.0" ?>
<! DOCTYPE Is SYSTEM "Is.DTD">
<Is>
<Rule Type="Object" Show="Replace" Href="http://127.0.0.1/fis/cgi-bin/MasterWrapper.asp"
  ObjectID="FIS,FL,EBIT-EPS,Y1998,A3" ObjectType="Cell"
  Command="No">EBIT</Rule>
. . . .
</Is>
```

The object identifier (ObjectID) is the key to determine which object of the given system the command should operate on. To aid the reader in understanding our examples, the following table lists possible identifiers for FIS objects. For example, the ObjectID "FIS,FL,EBIT-EPS" means the "EBIT-EPS" workbook of the "FL" subsystem. FIS stands for financial information system.

We divide the Command_Routine into sub-procedures (see figure 4) which should provide the following functions:

(1) map commands marked in the wrappers to actual IS commands;

(2) send actual commands and other parameters (e.g., ObjectID) to the IS;

(3) receive output from the IS;

(4) convert dynamically generated information objects to <Rule> tags (i.e., hyperlinks);

(5) create the XML document with <Rule> tags and send the document to the Web server.

Here is an example in which the command accesses a structural relationship (from step 2). Command_Routine("ShowData", "FIS,FL,EBIT-EPS,Y1998", "Sheet") will execute the five aforementioned functions and send the XML document in table 7 to the WWW server. As an example, we just list the cell "EBIT" here.

This type of <Rule> tag is called an object <Rule> tag since the value of the "Type" attribute is "Object". In the object <Rule> tag, the value of the "Command" attribute is "No". When a user clicks on the <Rule> tag, the "MasterWrapper.asp" application will be invoked and then the Object_Routine will be called to infer available commands for this object.

In figures 5–10, we present the interface and some outputs of our prototype.

5. Future work

Our mapping routines convert commands and information objects to useful links that give users direct access to an ISs' primary functionality, give access to various relationships about IS objects, and enable annotation and ad hoc links. In this paper, we defined an XML DTD to demonstrate its feasibility for representing mapped information generated by information systems. We also described a prototype implementing it for a financial information system.

Future research could integrate our prototype with more information systems to examine the completeness of our DTD. Information systems have different information structures and relationships, so we may need to add more elements and attributes to our DTD to integrate with the larger universe of information systems.

We also plan to extend the mapping routines and XML DTD to support sophisticated navigation technique (e.g., guided tours, overviews, and structural query).

Our prototype uses XSL for displaying and manipulating links and just offers simple linking capability. We shall explore merging our XML DTD with the newly emerging Web standards for hypermedia linking: XML Linking Language (XLink) and XML Pointer Language (XPointer). XLink allows authors to create link elements, which represent relationships between two or more resources (e.g., files, images, and videos) [16]. XLink offers two types of links: simple and extended links. A simple link is the two-ended inline link. An extended link has more than two participating resources. XLink provides unidirectional and multidirectional links. A multidirectional link allows users to initiate the link from more than one of its participating resources. XLink allows links to be inline or out-of-line. An inline link resides within one of the resources it joins. An out-of-line link does not reside in any of the resources that it joins. XLink also supports linkbases, which are XML documents that contain extended, out-of-line links. XPointer is a mini-language for specifying precise location of sub-parts of an XML document [15]. In XLink, one can add a pound sign (#) or a bar (|) to the end of the Uniform Resource Identifier (URI) and then place the XPointer on the end of that. XLink and XPointer together will help a browser to jump to an XML document and scroll to the precise location within it.

Lastly, we intend to develop extensive guidelines for stages 1, 3, and 4 of our RNRA approach: identify objects, identify relationship commands and build mapping routines. These guidelines will enable designers and developers to implement an integrative architecture quickly.

6. Conclusion

This paper makes two contributions. First, we propose a procedure for analyzing ISs and building mapping routines. Second, we define an XML DTD that declares a set of elements and attributes for representing mapped information in a human-readable, machine-readable, structured, and semantic way.

The WWW provides an opportunity to integrate hypermedia into information systems. We believe that integrating information systems and ISs in the business world with the WWW should constitute a major thrust for the WWW research. It will go a long way toward making applications more understandable. When reengineering applications for the WWW, dynamic relationship mapping could be an effective way to add additional hypermedia links. This will facilitate adding useful hypertext functionality to new WWW applications (especially ISs) [7,8]. We hope this paper will call people's attention to this opportunity.

XML is becoming increasingly popular for transmitting data within WWW applications and has been advocated by the hypermedia and WWW research communities for implementing hypertext linking. Our research shows how to merge these two efforts as we progress towards the goal of giving hypermedia support to all information systems.

Acknowledgements

We gratefully acknowledge support for this research by the NASA JOVE faculty fellowship program, by the New Jersey Center for Multimedia Research, by the National Center for Transportation and Industrial Productivity at the New Jersey Institute of Technology (NJIT), by the New Jersey Department of Transportation, and by the New Jersey Commission of Science and Technology, and by Rutgers University.

Appendix A

Document type definition for representing IS information.[a]

```
<!– The top-level IS element –>
<! ELEMENT      Is                      (Rule | Rel | P) +>
<!– RULE element: represents an IS object or command for accessing various relationships on the
selected object. It could be displayed like a hyperlink. –>
<! ELEMENT      Rule                    (#PCDATA)>
<!– P element: represents a paragraph. –>
<! ELEMENT      P                       (#PCDATA)>
<!– REL element: a container for four elements representing relationships. –>
<! ELEMENT      Rel                     (Comment | Info | Component | Relationship)+>
<!– INFO element: represents metainformation. –>
<! ELEMENT      Info                    (#PCDATA) >
<!– COMPONENT element: represents an IS object. –>
<! ELEMENT      Component               (#PCDATA) >
<!– RELATIONSHIP element: represents a schema relationship between two IS objects. –>
<! ELEMENT      Relationship            (#PCDATA)
<!– COMMENT element: represents an annotation (i.e., comment) on an IS object. –>
<! ELEMENT      Comment                 (Subject | Author | Date | Content)>
<!– SUBJECT element: represents the subject of a comment. –>
<! ELEMENT      Subject                 (#PCDATA)>
<!– AUTHOR element: represents the author of a comment. –>
<! ELEMENT      Author                  (#PCDATA)>
```

(Continued on next page)

(Continued.)

<!– DATE element: represents the date of creating a comment. –>		
<! ELEMENT	Date	(#PCDATA)>
<!– CONTENT element: represents the content of a comment. –>		
<! ELEMENT	Content	(#PCDATA)>
<!– A list of attribute declarations for the Rule element. –>		
<! ATTLIST	Rule	
Type	(Object | Command)	Object
Href	CDATA	#REQUIRED
ObjectID	CDATA	#REQUIRED
ObjectType	CDATA	#REQUIRED
Show	(Embed | Replace | New)	"REPLACE"
Command	(#PCDATA)	#REQUIRED >
<! ATTLIST	Rel	
ObjectID	CDATA	#REQUIRED
Type	(Schema | Structural | Metainformation | Annotative) #REQUIRED>	
<! ATTLIST	Info	
Type	(Text | Number | Formula)	#REQUIRED>
<! ATTLIST	Component	
Name	CDATA	#REQUIRED
Id	CDATA	#REQUIRED
Level	CDATA	#REQUIRED>
<! ATTLIST	Relationship	
Name	CDATA	#IMPLIED
From	CDATA	#REQUIRED
To	CDATA	#REQUIRED >

[a] Notes:

1. Element: A logical unit of information within an XML document.
2. Tag: A tag (e.g., <Rule>) marks an element (e.g., Rule) within an XML document. Some elements (e.g., Rule) must have start tag (e.g., <Rule>) and end tag (</Rule>).
3. "<!–" and "–>": Comments begin with <!– and end with –>.
4. "|": Indicates that one of a number of elements must be present.
5. "+": Indicates that an element or group of elements may appear one or more times.
6. "#": Indicates that the keyword that follows is a reserved word.
7. CDATA: Stands for "character data" and indicates that any ASCII text is allowed for this attribute value.
8. #PCDATA: Stands for "parsed character data" and indicates that only text is allowed inside an element.
9. #REQUIRED: Indicates that the attribute must appear in the start tag within the XML document.
10. #IMPLIED: Indicates that the attribute value is not required and no default value is provided.
11. ATTLIST: Indicates a list of attribute declarations for a particular element type.
12. Embed: Indicates that the destination document should be embedded in the current document when the link is traversed.
13. Replace: Indicates that the destination document should replace the current document when the link is traversed.
14. New: Indicates that the destination document should be displayed in a new window when the link is traversed.

References

[1] S. Adler, A. Berglund, J. Caruso, S. Deach, P. Grosso, E. Gutentag, A. Milowski, S. Parnell, J. Richman and S. Zilles, Extensible Stylesheet Language (XSL) Version 1.0, http://www.w3.org/TR/xsl/ (2000).

[2] R. Akscyn, D. McCracken and E. Yoder, KMS: A distributed hypermedia system for managing knowledge in organizations, Communications of the ACM 31 (1988) 820–835.
[3] C. Allen, WIDL: Application integration with XML, XML Special Issue of the World Wide Web Journal 2 (1997).
[4] M. Bieber, Automating hypermedia for decision support, Hypermedia 4 (1992) 83–110.
[5] M. Bieber, Hypertext and Web engineering, in: *Proceedings of Hypertext'98* (1998) pp. 277–278.
[6] M. Bieber and C. Kacmar, Designing hypertext support for computational applications, Communication of the ACM 38 (1995) 99–107.
[7] M. Bieber, H. Oinas-Kukkonen and V. Balasubramanian, Hypertext functionality, ACM Computing Surveys (forthcoming).
[8] M. Bieber and F. Vitali, Toward support for hypermedia on the World Wide WWW, IEEE Computer 30 (1997) 62–70.
[9] M. Bieber, F. Vitali, H. Ashman, V. Balasubramanian and H. Oinas-Kukkonen, Fourth generation hypermedia: Some missing links for the World Wide Web, International Journal of Human Computer Studies 47 (1997) 31–65.
[10] T. Bray, RDF and Metadata, http://www.xml.com/xml/pub/98/06/rdf.html (1998).
[11] T. Bray, J. Paoli and C.M. Sperberg-McQueen, Extensible Markup Language (XML). 1.0, http://www.w3.org/TR/1998/REC-xml-19980210 (1998).
[12] J. Bosak, XML, Java, and the future of the Web, XML Special Issue of the World Wide Web Journal 2 (1997).
[13] T. Catlin, P. Bush and N. Yankelovich, InterNote: Extending a hypermedia framework to support annotative collaboration, in: *Proceedings of Hypertext'89* (1989) pp. 365–378.
[14] C.M. Chiu and M. Bieber, A dynamically mapped open hypermedia system framework for integrating information systems, Information and Software Technology (forthcoming).
[15] S. DeRose, R. Daniel and E. Maler, XML Pointer Language (XPointer), http://www.w3.org/TR/xptr (1999).
[16] S. DeRose, E. Maler, D. Orchard and B. Trafford, XML Linking Language (XLink), http://www.w3.org/TR/xlink/ (1999).
[17] G.H. Collier, Thoth-II: Hypertext with explicit semantics, in: *Proceedings of Hypertext'87* (1987) pp. 269–290.
[18] J. Conklin, Hypertext: An introduction and survey, IEEE Computer 20 (1987) 17–41.
[19] D. Connolly, R. Khare and A. Rifkin, The evolution of the Web documents: The ascent of XML, XML Special Issue of the World Wide Web Journal 2 (1997) 119–128.
[20] G.W. Furnas, Generalized Fisheye Views, in: *Proceedings of the ACM CHI'86 Conference on Human Factors in Computing Systems* (1986) pp. 16–23.
[21] A. Garrido and G. Rossi, A framework for extending object-oriented applications with hypermedia functionality, The New Review of Hypermedia and Multimedia 2 (1996) 25–41.
[22] F. Halasz, Reflection on NoteCards: Seven issues for the next generation of hypermedia systems, Communications of the ACM 31 (1988) 836–855.
[23] R. Khare and A. Rifkin, X marks the spot: Using XML to automate the Web, IEEE Internet Computing 1 (1997) 78–87.
[24] O. Lassila and R.R. Swick, Resource Description Framework (RDF) model and syntax specification, http://www.w3.org/TR/REC-rdf-syntax (1999).
[25] Y.K. Lee, S.-J. Yoo, K. Yoon and P.B. Berra, Querying structured hyperdocuments, in: *Proceedings of the 29th Annual Hawaii International Conference on System Sciences* (1996) pp. 155–164.
[26] J. Nielsen, The art of navigating through hypertext, Communications of the ACM 33 (1990) 296–310.
[27] J. Rosenberg, The structure of hypertext activity, in: *Proceedings of Hypertext'96* (1996) pp. 22–30.
[28] R. Trigg, Guided tours and tabletops: Tools for communicating in a hypertext environment, ACM Transactions of Office Information Systems 6 (1988) 398–414.
[29] K. Utting and N. Yankelovich, Context and orientation in hypermedia networks, ACM Transactions on Information Systems 7 (1989) 58–84.

[30] J. Yoo, Relationship analysis, Ph.D. thesis, Rutgers University, Newark, NJ (2000).
[31] J. Yoo and M. Bieber, Towards a relationship navigation analysis, in: *Proceedings of the 33rd Hawaii International Conference on System Sciences* (2000).
[32] J. Yoo and M. Bieber, Finding linking opportunities through relationship-based analysis, in: *Hypertext '00 Proceedings* (2000) pp. 181–190.