ELSEVIER

Information and Software Technology 43 (2001) 75–86

INFORMATION AND SOFTWARE TECHNOLOGY

www.elsevier.nl/locate/infsof

A dynamically mapped open hypermedia system framework for integrating information systems

C.-M. Chiu[a,*], M. Bieber[b]

[a]Department of Information Management, National Kaohsiung First University of Science and Technology, University Rd. Yenchao, Kaohsiung, Taiwan, ROC
[b]Collaborative Hypermedia Research Lab, Computer and Information Science Department, New Jersey Institute of Technology, Newark, NJ 07102, USA

Received 13 May 1999; revised 3 September 1999; accepted 20 June 2000

Abstract

The overall goal of this research is to design a distributed, extensible, cross-platform, collaorative, and integrated system that can supplement information systems with hypertext support. In this paper, we propose a dynamically mapped open hypermedia system (DMOHS) framework that can support information systems fully. Our framework has two axes: a logical component focus and an application requirement focus. In Axis 1, we propose a conceptual DMOHS architecture with eight logical components. In Axis 2, we define and discuss major aspects of a DMOHS that should be supported in a distributed and integrated environment. Together the two axes provide a grid for specifying the logical DHOMS functionality for supporting application requirements. Given this framework, we first evaluate five open hypermedia systems and the www, and then design our own system implemented on top of the www. This paper also contributes guidelines for building mapping routines that supplement on top of the www. Further, it contributes guidelines for building mapping routines that supplement information systems with hypertext support. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Dynamic mapping; Hypermedia; Open hypermedia systems; Information systems; Mapping mechanism; World wide web

1. Introduction

Many information systems (IS), both on and off the World Wide Web (www), do not take full advantage of linking and navigation. This occurs for several reasons. In part, it has not occurred to many designers and developers to incorporate hypermedia functionality (i.e. an extremely rich layer of links and other navigation, structuring and annotation functionality) [5]. Most designers and developers do not have a hypermedia mindset; they and their users have seen few examples and do not demand this functionality as yet. In part, people do not have the time to re-engineer existing applications, especially when migrating them to the www, incorporating hypermedia functionality, easily. Developers will not do this until it is natural to conceive and easy to implement. These reasons constitute much of the motivation for our research.

A major goal of this paper is to present a framework for integrating hypermedia functionality into the everyday IS that people use.

We view hypertext as value-added support functionality [4]. Hypertext structuring, annotation and navigational functionality can enrich business, scientific, engineering and personal applications, thereby making them more effective. People use these applications primarily for their underlying analytical functionality, i.e. not for their ability to produce hypertext documents or displays. People will not abandon the applications they use everyday in favor of those that offer hypertext. Therefore, we need to bring hypertext support to these applications. Augmenting applications with direct access and hypertext structuring, annotation and navigation functionality [3,5] should result in new ways to: view an application's knowledge and processes conceptually; navigate among items of interest and task stages; enhance an application's knowledge with comments and relationships; and target information displays to individual users and their tasks.

So far there is no precise definition or requirements for an open hypermedia system. Davis et al. [12] believe a reasonable definition for an open hypermedia system should be as follows: (1) the system should be able to contain hypermedia objects to the size of the objects or to the maximum number of such objects; (2) the system should allow the import and use of any data format; (3) the system should be configurable and extensible to incorporate new hypermedia mode; (4) the system should be able to distribute

* Corresponding author. Tel.: +886-7-6011000; fax: +886-7-6011042.
E-mail addresses: cmchiu@ccms.nkfu.edu.tw (C.-M. Chiu), bieber@njit.edu (M. Bieber).

data and processes across a network and across hardware platforms; and (5) the system should support multiple users.

While OHS provide a starting set of integration tools for providing IS with hypermedia functionality, they primarily serve static applications with manually invoked hypermedia features. Integrating with an OHS generally requires changes to the IS being integrated. Many IS applications, however, are not static; they generate their documents and display screens dynamically in response to user queries, or as they "push" information to user screens. Developers could enrich these dynamically generated displays if they could pre-specify links and other hypermedia features to generate dynamically along with the displays, instead of only allowing users to add hypermedia features manually. (While some database applications do this, the links they specify are rather limited in comparison to what we envision.) Lastly, as it would be impractical to modify many existing IS applications, we are striving to provide hypermedia functionality with little or no change to the applications.

The overall goal of this research is to design a dynamically mapped open hypermedia system (DMOHS) — a distributed, extensible, cross-platform, collaborative, and integrated architecture that can supplement other information systems, with hypertext support. We divide the research process into three steps. First, we propose a DMOHS framework that can support information systems (IS) fully. Second, we use the framework to evaluate five open hypermedia systems and the www, to decide which ones support our prototype's research goals. Third, we discuss how to build dynamic-mapping routines and then implement a prototype that can benefit www users.

This research makes three major contributions. First, no OHS supports IS applications with dynamically generated hypermedia functionality without altering the application. Our framework makes the requirements for such integration explicitly. Second, we model the mapping routines that specify the hypermedia links to generate dynamically. Third, we demonstrate the use of the framework with a dynamically mapped open hypermedia system prototype that meets many of the framework's requirements.

Our DMOHS framework has two axes: a logical component focus and an application requirement focus (see Fig. 1). The first encompasses a conceptual DMOHS architecture. The second highlights the important features that a DMOHS should support. Given this framework (see Fig. 1), developers can fill in the functional specification for each component within our conceptual architecture. With a complete functional specification developers can start to build their own systems from scratch or with the help of existing systems.

We use this DMOHS framework to evaluate five open hypermedia systems and the www, to decide which can best provide information systems with hypermedia support. We use these systems for two main reasons: (1) the current work in providing hypertext to applications has been explored primarily in the context of open hypermedia systems (OHSs) [10,21]; and (2) the www is the most successful and widely used distributed hypermedia system. We note that both OHSs and the www support distribution, extensibility, integration, cross-platform functionality to some extent. Our evaluation results (see Section 3) show that the www is superior to open hypermedia systems in features like distribution, extensibility, cross-platform, and availability of resources (e.g. servers and browsers).

In order to provide hypertext functionality to hypertext-unaware information systems (IS) with minimal changes to them, we propose to use wrappers and mapping routines. ISs dynamically generate information to display and thus require some *mapping mechanism* to automatically map the generated content to hypertext constructs (e.g. nodes, links, and link markers). This mapping mechanism infers useful links that give users direct access to the ISs' primary functionality, give access to meta-information about IS objects, guide users to browse information objects, and enable annotation and ad hoc links. This paper contributes a systematic dynamic-mapping mechanism for mapping outputs from an information system to hypertext constructs.

This paper is organized as follows. In Section 2 we present the DMOHS framework. In Section 3 we apply the framework. In Section 3 we apply the framework to each OHS system and the www. Section 4 discusses how to build mapping routines and presents our prototype. In Section 5 we discuss future research plans and there is a general discussion of the issues. Section 6 concludes with a larger view of our research.

2. The DMOHS framework

The first axis of our framework provides a comprehensive viewpoint for thinking about a DMOHS architecture that integrates ISs. The second axis provides a set of major application requirements that a DMOHS should support. The DMOHS framework extends OHS components and features. This framework emphasizes the integration with ISs, providing hypertext functionality to them. We add back-end applications (i.e. ISs) to OHS components and integration with ISs to OHS features to form Axes 1 and 2 separately. In Sections 2.1 and 2.2 we will discuss Axes 1 and 2 in more detail.

Users will wish to use existing third party applications that run under their host operating systems, to create documents and access documents on remote machines. So, we believe that *integration*, *distribution*, and *cross platform support* constitute the major requirements of OHSs [10,16]. We have incorporated these three requirements into Axis 2 of our DMOHS framework. Malcolm et al. [15] assert that hypermedia systems must provide a distributed heterogeneous environment to allow users to create and modify information *collaboratively*. Open hypermedia systems should provide an extensible architecture to work

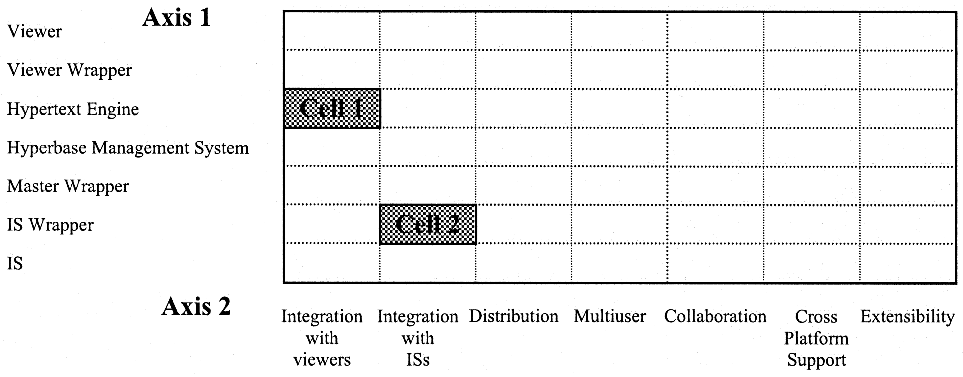


Fig. 1. Dynamically mapped open hypermedia system framework.

with new viewers, ISs, data formats, and protocols [18,21]. Therefore, we included these requirements in Axis 2 of our framework as well.

The two axes of our framework are orthogonal. For example, Fig. 1, cell 1 shows that, to enable the integration of hypertext with viewers (Axis 2) the hypertext engine (Axis 1) should provide certain functionality, such as sending requests to and receiving requests from the viewer. Such specifications would be included within cell 1. Cell 2 shows that to integrate hypertext with ISs (Axis 2), IS wrappers (Axis 1) should provide certain functionality such as mapping routines to map display outputs from ISs to hypertext constructs (e.g. nodes, links, and link markers).

We note that one can analyze systems using each axis independently or using both together. Together the two axes provide a grid for specifying the logical DMOHS functionality for supporting application requirements. Our framework provides a way to analyze both existing and proposed OHSs.

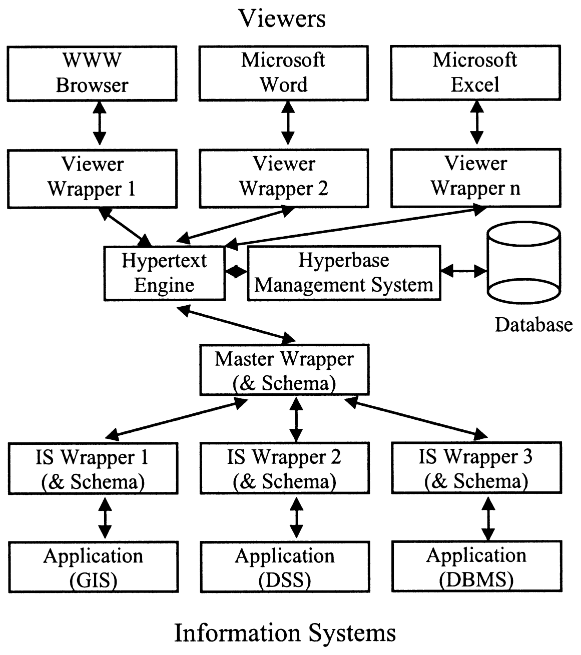


Fig. 2. A conceptual DMOHS architecture integrating example viewers and information systems.

So far, only two frameworks exist for classifying and evaluating hypermedia systems: the Flag taxonomy and our DMOHS framework. The Flag taxonomy allows one to: (1) classify existing hypermedia systems, (2) characterize what an open hypermedia system is, and (3) examine (describe and compare) OHSs independent of the particularities of specific systems [17]. A major difference between our DMOHS framework and Flag taxonomy is that our framework explicitly distinguishes storage-level components (i.e. information systems) from presentation-level applications (i.e. viewers). Our DMOHS framework also makes many of the integration requirements more explicit, both in terms of Axis 1's architectural components and Axis 2's functionalities. While DMOHS is a general framework, it does advocate a particular approach to integration, i.e. through a hypermedia engine that uses application schemata to map hypermedia functionality dynamically. We believe our dynamically-mapped OHS framework is more extensible than the Flag taxonomy, because we can remove items from or add items to the two axes to form appropriate sub-frameworks to evaluate new hypermedia systems or new features of existing hypermedia systems. We hope that our framework gives people a view of an open hypermedia system in a simpler and clearer way than the flag taxonomy.

2.1. Axis 1: DMOHS logical component focus

In this axis, we propose a conceptual DMOHS architecture with eight logical components. This architecture emphasizes integration with both viewers and ISs, providing hypertext functionality to each. Fig. 2 sketches examples of this architecture. Components communicate with each other through message passing. A major research issue concerns what functionality each component logically should provide. Note that while actual DMOHSs will include some or all of this functionality, their architectures may

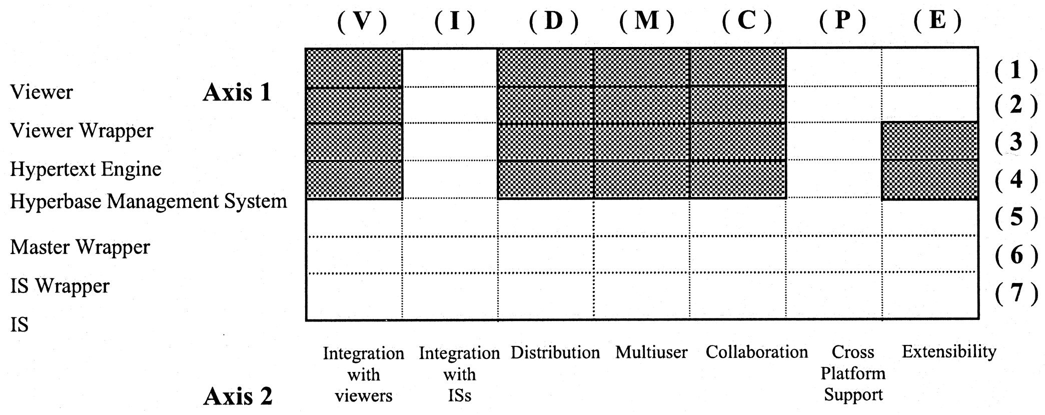


Fig. 3. Evaluation of HyperDisco.

physically implement the functionality in different modules than those shown here.

1. An IS is an application system with which users interact to perform some task, which often dynamically produces output content for display.
2. An IS wrapper translates and routes messages between its IS and the hypertext engine. It also provides its IS's mapping schema. A comprehensive IS wrapper will allow us to integrate an existing IS with few or no changes. An IS wrapper is responsible also for performing any task required by the engine for compliance, but which the IS itself actually cannot do.
3. The master wrapper coordinates schema mapping among different IS domains, thus aiding IS-to-IS integration.
4. The hyperbase management system (HBMS) maintains the structure and document contents stored in the hypertext database. As we will state later, link server-based systems use viewers to store document contents and open hyperbase-based systems use either the HBMS or viewers to store document contents. In a multiuser/collaborative environment, the HBMS should provide functionality such as concurrency control, access control, notification or versioning control.
5. The hypertext engine provides and controls hypertext functionality for both IS and viewer applications and maps hypertext components to displays of the IS information. The responsibilities of the hypertext engine include: keeping records of users' activities, invoking the viewer through its wrapper, resolving link activation, supporting navigation aids, information filtering, coordinating with the HBMS to access hypertext components through the communication protocol, and coordinating message passing in a distributed and collaborative environment.
6. A viewer wrapper translates and routes messages between its viewer and the hypertext engine [20]. A viewer wrapper should provide a communication protocol for a viewer and the hypertext engine to exchange messages. Existing protocols include DDE, OLE, TCP/IP sockets, inter-process communication, or Apple Events. A viewer wrapper is responsible also for performing any task required by the engine for compliance but which the viewer itself actually cannot do.
7. Viewers represent applications that can both display and edit hypertext components, in addition to their original functionality. This architecture also requires an appropriate viewer to display the IS output. In other words, the hypertext engine will invoke an appropriate viewer that can handle the output from an IS. The responsibilities of viewers include displaying and editing documents, displaying link markers, handling anchor values, providing menus that support hypertext functionality, allowing users to highlight the content of documents, storing contents (for link server-based systems), and communicating with the hypertext engine.
8. The database stores links and documents contents.

2.2. Axis 2: application requirement focus

In this axis, we define and discuss aspects of DMOHS that must be supported in a distributed and integrated environment. We believe that integration, distribution, cross-platform support, and extensibility constitute the most important requirements of a DMOHS. Axis 2 helps us analyze these, as well as other requirements for our conceptual architecture. Within the full double-axis grid (Fig. 1), we can specify then the functionality each logical component of our architecture could provide to support application requirements.

1. Integration with viewers: integrating viewers and their functionality into a DMOHS and providing manual hypertext support.
2. Integration with ISs: integrating ISs and their functionality into a DMOHS and providing hypertext support for dynamically-generated content.
3. Distribution: distributing all eight logical components across a network. In a distributed environment, communication among system components is a main issue.

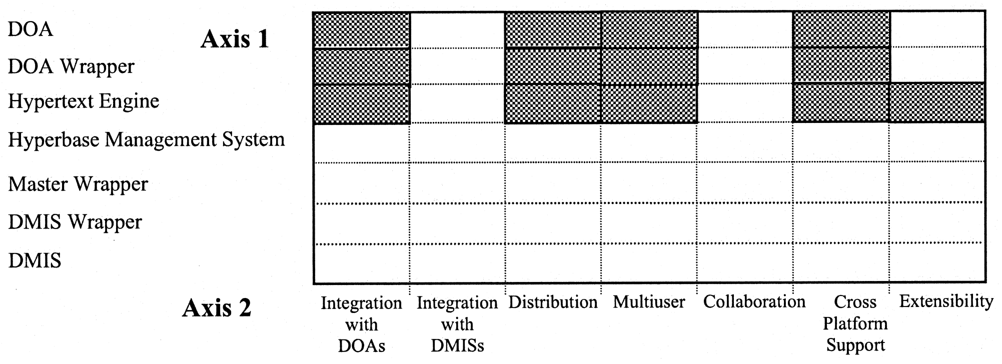


Fig. 4. Evaluation of Microcosm.

4. Multiuser: allowing multiple users to access documents under concurrency and access control.
5. Collaboration: support for collaborative environments. Multiuser hypermedia systems are different from collaborative hypermedia systems. Multiuser systems provide access and concurrency control. Collaborative systems provide those two functions and more notification controls than multiuser systems, so that multiple users can work together.
6. Cross-platform support: support for distribution across heterogeneous operating systems. A DMOHS should allow a user to access information across the network and heterogeneous operating systems (e.g. MS-Windows, UNIX, Macintosh).
7. Extensibility: whether new components (e.g. ISs) and data formats can be added to the conceptual architecture and whether new functionality can be added to components.

3. Evaluation

We have evaluated five open hypermedia systems and the www with the DMOHS framework, to decide which one, if any, can help us build a prototype covering the full breadth of the framework. In this paper we present the detailed evaluation results of three systems.

3.1. Evaluation of HyperDisco

Fig. 3 shows our evaluation of HyperDisco. The shaded areas in the figure indicate that HyperDisco has logical components equivalent to those in Axis 1 for supporting the application requirements of Axis 2.

- HyperDisco was developed by Uffe Kock wiil at Aarhus University, Denmark. HyperDisco integrates with tools (viewers), which both comply fully with its protocol and which are non-compliant (see Fig. 3, cell V1). HyperDisco's tool integrator serves a similar role to the viewer wrapper and the hypertext engine in our architecture combined (see cells V2 and V3). The tool integrator's integration model layer provides a basic hypermedia linking service (anchors and links). The HBMS can handle the storage of hypertext contents or let the viewers handle content storage (see cell V4).
- The survey result show that HyperDisco can integrate with ISs either through an extensible and tailorable communication protocol (e.g. based on sockets) or the Scheme language. HyperDisco uses the programming language Scheme extended with standard object-oriented features as its scripting language. HyperDisco has an operating system interface and can perform the same operations as www CGI scripts. We do not put any shaded area on the second column because HyperDisco does not integrate with any wrapper to infer useful links that give users more direct access to the IS;s primary functionality, give access to meta-information about IS objects, or enable annotation and ad hock links (see cells I1 to 17). ISs dynamically generate their contents and thus require some *mapping mechanism* to automatically map the generated content to hypertext constructs (nodes, links, ad link markers).
- HyperDisco works in the WAN environment by using a two-level name service mapping schema [22] (see cell D1 to D4).
- HyperDisco provides short database transactions combined with user-controlled locking to allow multiple users to work simultaneously without damaging data integrity (see cells M1 to M4).
- HyperDisco can support both asynchronous and synchronous collaboration. A participating tool (viewer) needs a user interface to initiate an explicit lock request. Event notifications from HyperDisco enable viewers to indicate changes in the shared document network [21] (see cells C1 to C4).
- HyperDisco does not support multiple platforms. It only runs on Sun SparcStations (see cells P1 to P7).
- HyperDisco provides two layers of hypermedia functionality: An integration model layer and data model layer. In the integration model layer, the built-in classes provide basic hypermedia linking services (anchors and links) including a flexible communication protocol (see cell E3). In the data model layer, the built-in classes provide

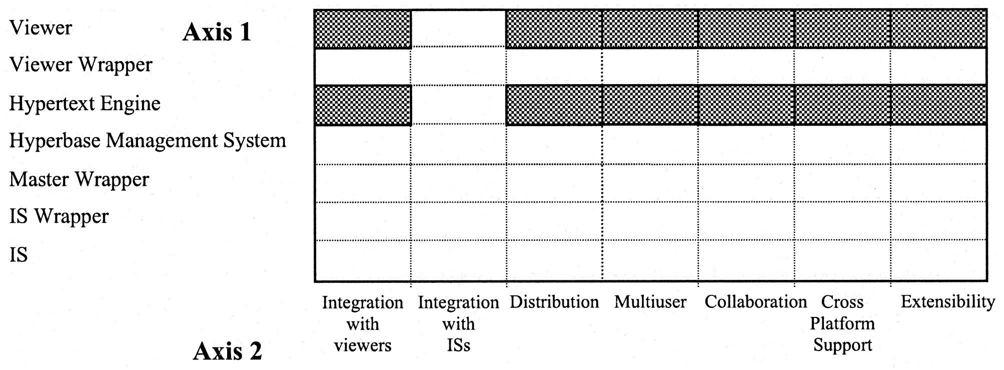


Fig. 5. Evaluation of the www.

basic hypermedia storage services for hypermedia objects (nodes, composites, links, and anchors) [21] (see cell E4). Both layers can be extended and tailored using (multiple) inheritance to provide specialized integration models for individual tools or tool groups.

3.2. Evaluation of Microcosm (Fig. 4)

- Microcosm was developed at the Image and Multimedia Research Group at the University of Southampton, England. Microcosm can integrate with DOAs at the fully-compliant, partially-compliant, and non-compliant levels [10,11]. Microcosm's architecture contains a central document control system, functional modules (called "filters"), and a link dispatcher [10]. We could treat these three components together as the hypertext engine. Microcosm uses a linkbase to store link information and DOAs to store node contents. Actually the linkbase is one of its filters.
- Microcosm itself does not support integration with DMISs. It can be coerced to do so by adding a new filter that serves as a DMIS wrapper.
- Microcosm's released version runs with data distributed across the LAN or WAN (but all processes run on the client machine). A number of lab versions run a true WAN distribution. Microcosm works in the WAN environment by integrating with the www. It adopts the so-called distributed link service (DLS) [8]. The DLS allows users to add links to the www document without writing link tags. The link information will be stored in linkbases available on the Web. There are two components of DLS: the client interface and the link server. The server is implemented using CGI scripts and is accessed through a Web server. The client side works by integrating with Netscape's Navigator.
- Microcosm does not provide concurrency control. Users may share resources without the right to update them. Users can maintain their own copies of document and links that will need updating.
- The DLS is available for MS-Windows and UNIX, so users can access information across these two platforms.
- Functional filters of Microcosm provide major hypertext functionality. System developers can extend Microcosm by dynamically installing, removing or reordering the filters.

3.3. Evaluation of the www (Fig. 5)

- The Web browsers (e.g. Netscape's Navigator and Microsoft's Internet Explorer) resemble the role of viewers and integrate with Web servers at the fully compliant level. Some browsers (e.g. Netscape's Navigator) allow a user to edit current document at run time. The www requires formats such as HTML, GIF and JPEG, which prevent the documents being accessible to some third-party applications. This feature limits the possibility of integrating other third-party applications as viewers unless they can display HTML documents, although this is becoming increasingly prevalent. As a Web browser can communicate with a www server, we do not need a viewer wrapper. The www uses the file system instead of a HBMS.
- Web database development is a hot new field. Most database applications handle queries and generate HTML from query results without inferring links. These database applications do not meet our definition of integration with ISs since they do not infer useful inks that give users direct access to various information about database objects (e.g. meta-information of database tables). The www has the opportunity to integrate hypertext support into ISs. There are many potential approaches for integrating Web servers with information systems: traditional Common Gateway Interface (CGI), server APIs (e.g. Netscape's NSAPI and Microsoft's ISAPI), server-side Java, Active Server Pages (ASP), etc. No systematic approach exists, however, for integrating an IS (or analytical information system) with the www and giving users direct access to its interrelationships. To dynamically generate HTML documents with valuable inferred links we need a more generic mapping mechanism.
- To support internal objects, IS wrappers have to map display output from ISs to HTML documents with

Table 1
The summarized evaluation results of six systems

Systems requirements	www	HyperDisco	DHM	Microcosm	HOSS	Chimera
Integration with viewers	Yes	Yes	Yes	Yes	Yes	Yes
Integration with ISs	No	No	No	No	No	No
Distribution	Yes: WAN/LAN	Yes: WAN/LAN	Yes: LAN; WAN (through www)	Yes: LAN; WAN (through www)	Yes: LAN/WAN	Yes: LAN; WAN (through www)
Multiuser	Yes	Yes	Yes	Yes	Yes	Yes
Collaboration	Yes	Yes	Yes	No	No	Yes
Cross-platform	Yes: at least three platforms	No	Yes: Unix; Macintosh; MS-Windows	Yes: Unix; MS-Windows	No	No
Extensibility	Yes through: traditional CGI; server APIs; ASP; etc.	Yes through: scripting language-Scheme	Yes through: APIs	Yes	Yes through: APIs	Yes

parameters (e.g. object ID, object type, owning systems, etc.) in the link anchor tags (<A >). When a user clicks on a link the IS wrapper will extract parameters within the link and execute the command.

- The www is a successfully distributed hypermedia system. Web servers and browsers can be distributed on multiple platforms (e.g. UNIX, MS-Windows, Macintosh). Users access information across the network and platforms through Web servers and browsers.
- The www allows multiple users to access information. However, it does not support distributed authoring yet.
- Web collaborative systems (e.g. Netscape's Communicator and Microsoft's NetMeeting) allow users to exchange information, develop ideas, see or talk to people, share applications, or simply transfer files to one or more participants. However, current Web collaborative tools do not provide concurrency and access control to maintain data integrity for distributed authoring and they use special-purpose collaborative serves instead of Web servers.
- A user can use a CGI script or ASP program to extend the server's features to some extent. HTML only supports a fixed and limited tagsset. XML addresses the limitations of HTML. XML [7] is a subset of standard Generalized Markup Language (SGML). XML aims to develop machine-readable, human-readable, structured, and semantic documents that can be delivered over the Internet [9,14]. An important feature of XML is that it is extensible [6]. XML allows the development of custom and domain-specific elements and attributes.

Table 1 summarizes our evaluation of all five open hypermedia systems and the www. From the evaluation results, we find that integration with IS, distribution, collaboration, cross-platform support are still not well supported in current OHSs. Right now, Chimera [1], DHM [13], Microcosm [8], HyperDisco [22], ad HOSS [16] can work in the WAN environment. However, the first three systems support this feature through integrating with the www. Most OHSs themselves do not actually provide robust infrastructures that allow system components (e.g. viewers, HBMS, linkbases, and hypertext engines) to work together in a wide area and distributed environment. HyperDisco, Chimera and HOSS only work on one platform (Sun Sparcstation). However, other systems work at least on two platforms. The www is superior to open hypermedia systems regarding features like distribution, extensibility, and cross-platform; so we decided to implement our prototype by incorporating the www.

Availability of resources and the approaches for integrating information systems are also two important factors to consider, for us to decide whether to implement the prototype on top of the www. Many Web servers and browsers for various platforms are available on the Internet. Many approaches exist for integrating information systems into the www, such as Common Gateway Interface (CGI), Active Server Pages (ASP), server-side java, and server APIs. We can choose approaches and languages with which we are familiar to implement the prototype. However, open hypermedia systems only support a limited set of approaches and languages for integrating with external resources.

4. Implementation

The basic concept underlying our proposal is the use of mapping routines to automatically provide hypertext functionality to ISs when integrating them with the www.The www server will serve any IS application that has an appropriate wrapper. To integrate a new IS with the www, one has to build a wrapper and store information (e.g. commands) in the knowledge base. Thus, to provide an IS application with hypertext support, the developer has to declare mapping routines in the master wrapper and its IS wrapper. Note that one set of mapping routines can serve all instances of an IS. Mapping routines infer useful links that give users a more direct access to the IS's primary functionality, give access to meta-information about IS objects, and enable annotation and ad hoc links. In this section, we discuss how to build mapping routines that supplement ISs with hypertext support, and then present our prototype.

4.1. Building mapping routines

The main purpose of mapping routines is to infer useful links from the output dynamically generated by an IS, as well as commands for operating on IS objects. These routines reside in and become invoked by the master wrapper or IS wrapper. Note that one set of mapping routines can serve all instances of an IS. We identify two types of mapping routines: Command_Routine and Object_Routine. Bieber [2] and Wan [19] implemented mapping rules (or bridge laws) using Prolog. However, for clarity we discuss mapping routines in terms of functional procedural calls. We explain each by using our Microsoft-Excel-based prototype financial information system as the target IS.

4.1.1. Command_Routine (System, Type):

This routine infers commands for operating upon the selected object. The "System" parameter is used to discriminate among different ISs. This routine should provide the following functions.

- Search the knowledge database for commands accessing various relationships on the selected IS object.
- Map commands to links
- Form a HTML document that includes mapped links and send the document to the Web server.

For example, Command_Routine ("FIS", "SHEET") will execute the forementioned functions and create the following HTML document. We list some commands in the following table. In reality, the system would present all commands (or a filtered subset).

```
< HTML > <BODY >
…
< A href = "http://163.18.21.246/fis/MasterWrapper.asp?
SYSTEM = FIS&ID = FL,EBIT,Y1999&
TYPE = SHEET&COMMAND = "drawChart" >
drawChart < /A >
…
< A href = "http://163.18.21.246/fis/MasterWrapper.asp?
SYSTEM = FIS&ID = FL,EBIT,Y1999&
TYPE = SHEET&COMMAND = showData" >
showData  < /A >
…
< /BODY > </HTML >
```

In this example HTML document:

1. The ID "FL,EBIT,Y1999", means the "Y1999" worksheet of the "EBIT" workbook within the financial leverage subsystem (FL).
2. The IS wrapper is an ASP application called "Master-Wrapper". Note that the link anchor has a "Command" attribute with the value "showData". We call this type of link anchor a "command link anchor".

4.1.2. Object_Routine (Command, System, ID, Type):

This routine infers links from the output generated by the IS. The Object_Routine has four parameters. The object identifier (ID) is the key to determine which object of the given system the command should operate on. This routine should provide the following functions:

- Map commands to actual IS commands.
- Send actual commands and other parameters to the IS.
- Receive the display output from the IS.
- Infer links from the output generated by the IS.
- Create the HTML document with inferred links and send the document to the Web server.

Here is an example in which the command accesses a structural relationship. Object_Routine ("showData", "FIS", "FL,EBIT,Y1999", "SHEET") will execute the five forementioned functions and send the following HTML document to the Web server. We list some cells in the following table.

```
< HTML > <BODY >
…
< A href = "http://163.18.21.246/fis/MasterWrapper.asp?
SYSTEM = FIS&ID = FL,EBIT,Y1999,A3&
TYPE = CELL&COMMAND = NO" > EBIT < /A >
…
< A href = "http://163.18.21.246/fis/MasterWrapper.asp?
SYSTEM = FIS&ID = FL,EBIT,Y1999,C3&
TYPE = CELL&COMMAND = NO" > EPS < /A >
…
< /BODY > </HTML >
```

Note that the link anchor has a "Command" attribute the value of which is "NO". This indicates that it does not provide a command. We call this type of link anchor "object link anchor".

4.2. Prototype

We have created a proof-of-concept prototype (see Fig. 8) using a financial information system as the target IS. The financial information system was built using Microsoft Excel. The master wrapper is an ASP program (i.e. Master Wrapper.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. In this subsection, we

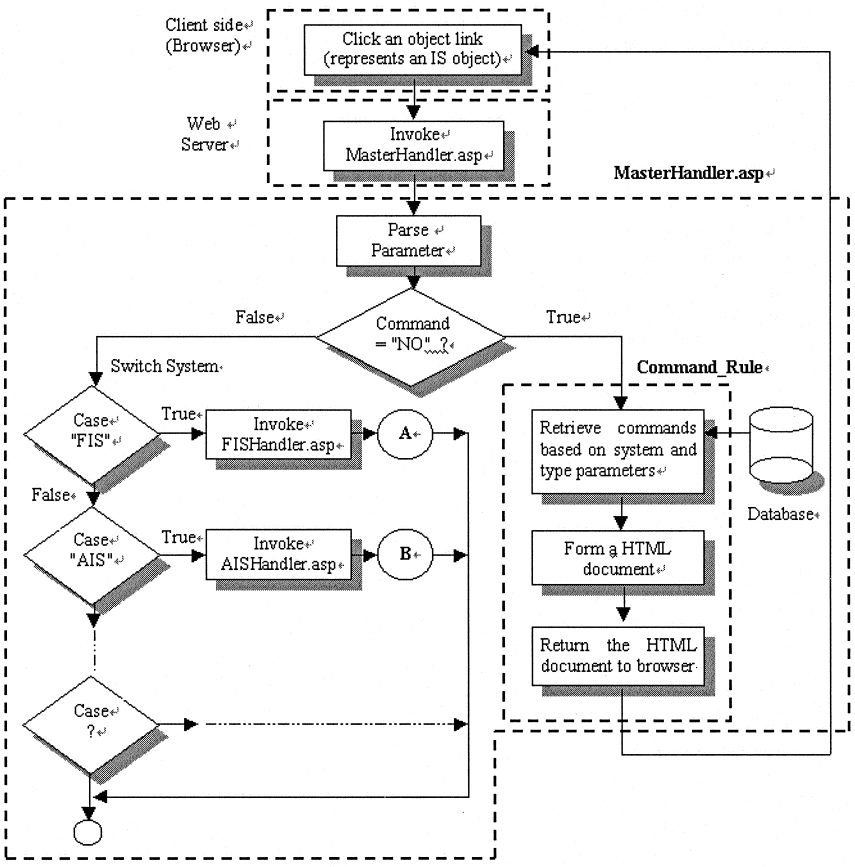


Fig. 6. The program flow of Master Wrapper.asp.

explain how to program Command_Routine and Object_Routine by using flowcharts (see Figs. 6 and 7).

Fig. 8 shows the interface of our prototype. Suppose that a user wants to retrieve the explanation of the terminology, "EPS". First, he clicks on the link labeled as "EPS". Mapping rules of the master wrapper will be invoked to infer available commands for the "EPS" object. These will replace those shown in Fig. 8. The command area of Fig. 8 shows the available command for the "EPS" object. Second, he clicks on the command link labeled as "explain". Mapping rules of the IS wrapper will be invoked to retrieve the explanation from the financial information system, form a HTML document, and then send the HTML document to the browser for display (see Fig. 9). Figs. 10–13 show outputs of executing other commands.

5. Future research

There are four directions for our future research. First, we are considering adding a third axis to ensure that a DMOHS design best serves applications in organizations. The third axis would include five traditional classes to the information system: transaction processing system (TPS), management information system (MIS), decision support system (DSS), group decision support systems (GDSS), and executive information systems (EIS). This axis would help us determine what requirements each of the logical components in Axis 1 has to fulfill to satisfy each class of information system. This axis also helps highlight which application requirements from Axis 2 are necessary for which class of IS.

Second, we will continue to improve the prototype in terms of the framework features not yet implemented, which include designing a mechanism for integrating with partially compliant and non-compliant viewers, integrating with multiple-information systems, and providing true Web collaborative functionality.

Third, we plan to develop guidelines, methodologies and components to help other OHS systems, especially those complying with the open hypermedia protocol, and provide more complete mapping with other OHS systems. Finally, we plan an implementation that covers the full framework.

6. Conclusion

The www and open hypermedia system provide opportunities to integrate hypermedia into the formation system. We believe that integrating information systems in the business world with the www and OHSs should constitute a major thrust for the www and OHS research. It will go a

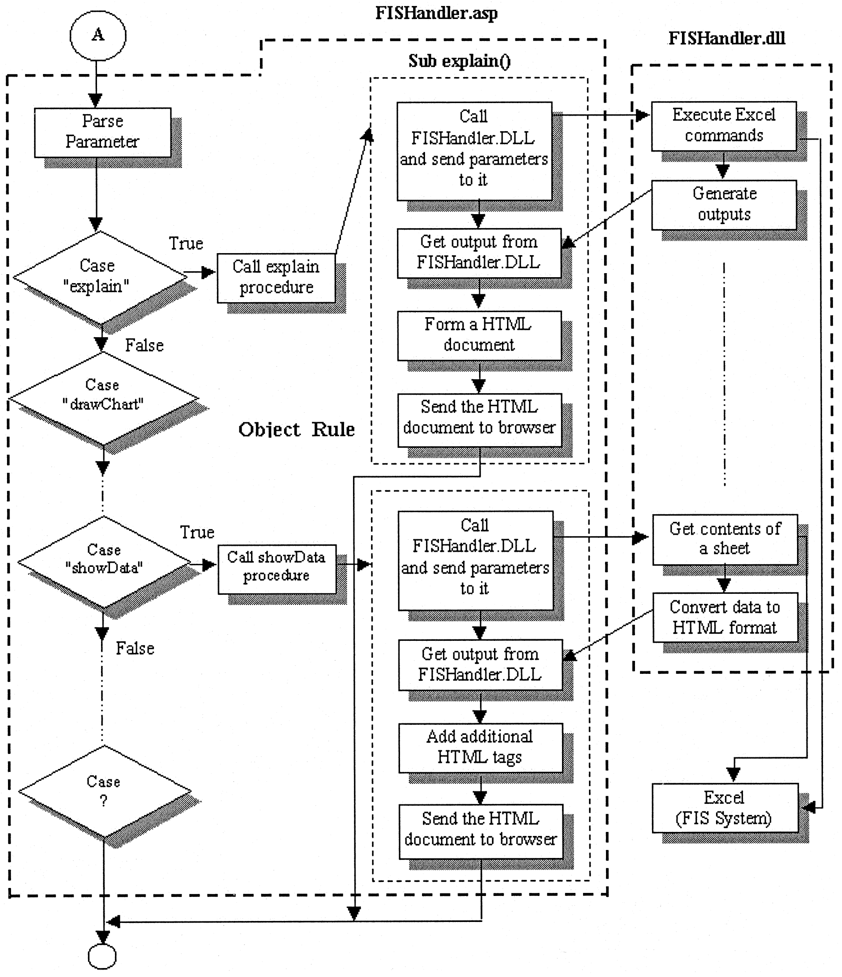
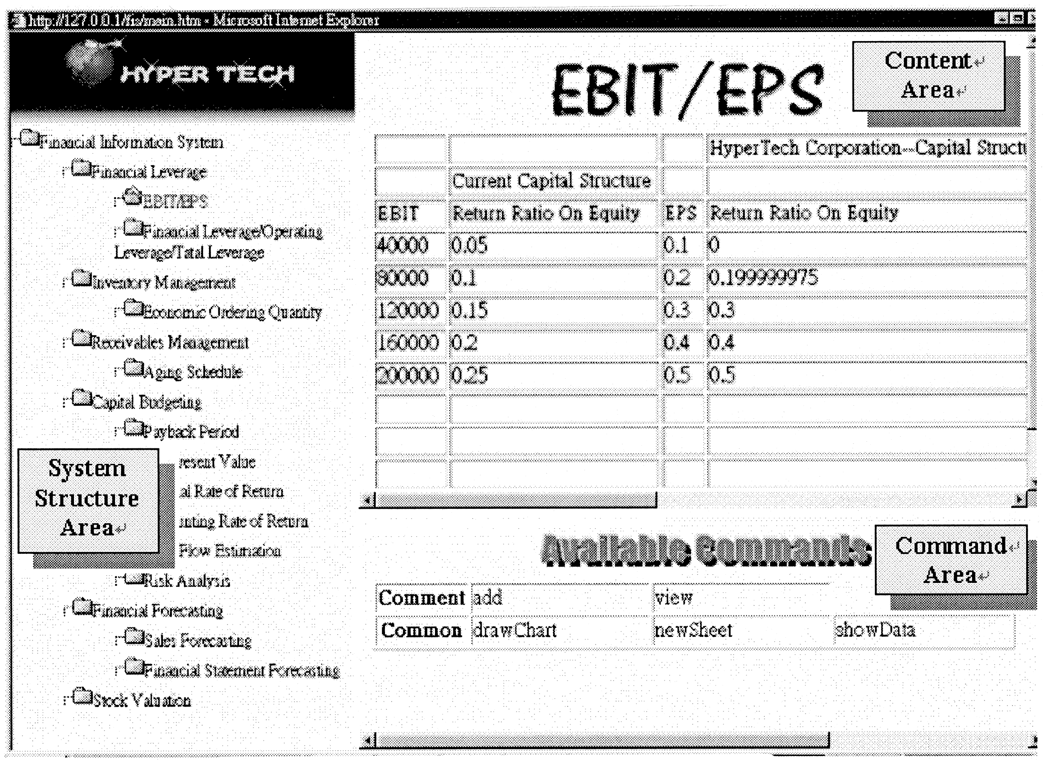


Fig. 7. The program flow of FISWrapper (i.e. FISWrapper.asp and FISWrapper.dll)


http://127.0.0.1/fis/main.htm - Microsoft Internet Explorer

HYPER TECH

- Financial Information System
  - Financial Leverage
    - EBIT/EPS
    - Financial Leverage/Operating Leverage/Total Leverage
  - Inventory Management
    - Economic Ordering Quantity
  - Receivables Management
    - Aging Schedule
  - Capital Budgeting
    - Payback Period
    - resent Value
    - al Rate of Return
    - nting Rate of Return
    - Flow Estimation
    - Risk Analysis
  - Financial Forecasting
    - Sales Forecasting
    - Financial Statement Forecasting
  - Stock Valuation

System Structure Area

EBIT/EPS

Content Area

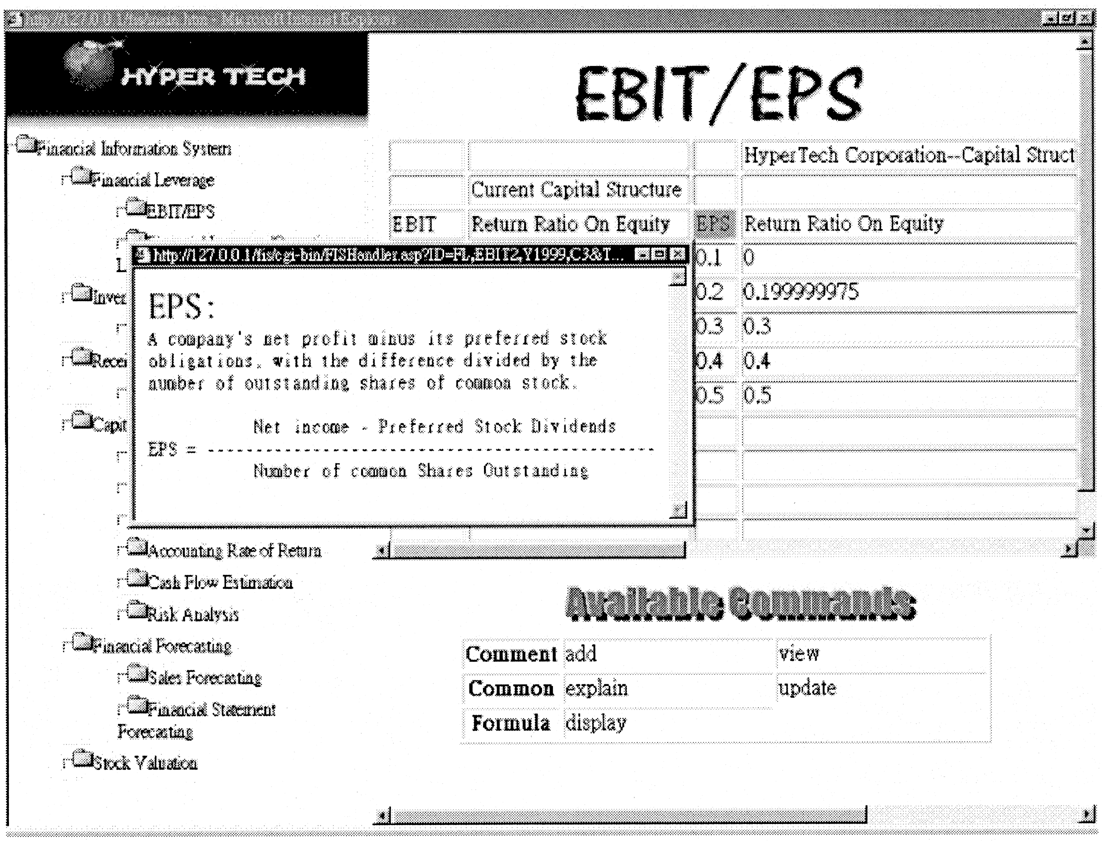
			HyperTech Corporation--Capital Structu
	Current Capital Structure		
EBIT	Return Ratio On Equity	EPS	Return Ratio On Equity
40000	0.05	0.1	0
80000	0.1	0.2	0.199999975
120000	0.15	0.3	0.3
160000	0.2	0.4	0.4
200000	0.25	0.5	0.5

Available Commands

Command Area

Comment	add	view	
Common	drawChart	newSheet	showData

Fig. 8. This figure show the interface of our prototype. This system has three areas: the system structure area, command area, and content area.



EBIT/EPS

			HyperTech Corporation--Capital Struct
	Current Capital Structure		
EBIT	Return Ratio On Equity	EPS	Return Ratio On Equity
		0.1	0
		0.2	0.199999975
		0.3	0.3
		0.4	0.4
		0.5	0.5

http://127.0.0.1/fis/cgi-bin/FISHandler.asp?ID=FL,EBIT2,Y1999,C3&T...

```
EPS:
A company's net profit minus its preferred stock
obligations, with the difference divided by the
number of outstanding shares of common stock.

              Net income - Preferred Stock Dividends
EPS = ---------------------------------------------
              Number of common Shares Outstanding
```

Comment	add	view
Common	explain	update
Formula	display	

Fig. 9. Output of executing the "explain" command.

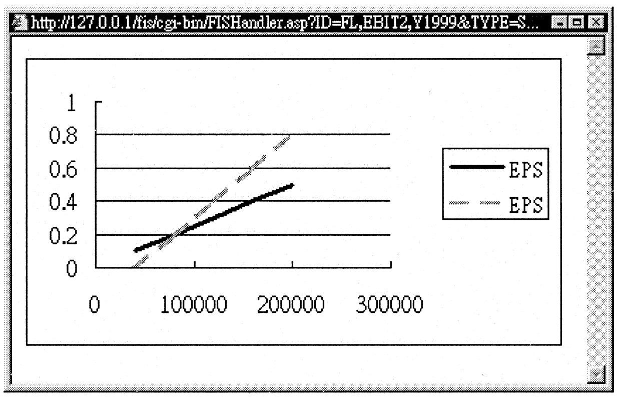
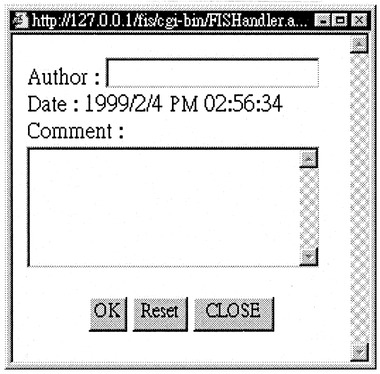
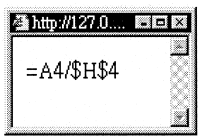


Fig. 10. Output of executing the "explain" command.



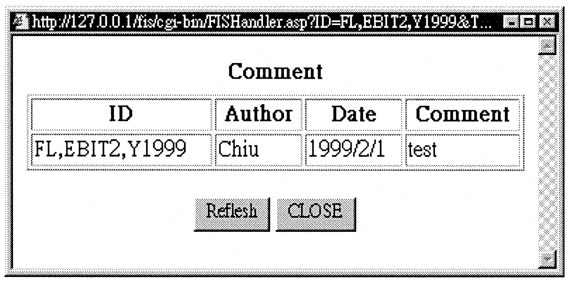
Author :
Date : 1999/2/4 PM 02:56:34
Comment :

Fig. 12. A form for users to add comments.



=A4/H4

Fig. 11. Output of executing the "view formula" command.


http://127.0.0.1/fis/cgi-bin/FISHandler.asp?ID=FL,EBIT2,Y1999&T...
Refresh
CLOSE

Comment

ID	Author	Date	Comment
FL,EBIT2,Y1999	Chiu	1999/2/1	test

Fig. 13. Output of executing the "view comment" command.

long way towards making applications more understandable. When re-engineering applications for the www and OHSs, dynamic mapping could be an effective way to add additional hypermedia links. This will facilitate adding useful hypertext functionality to new www and OHS applications (especially Iss). We hope this paper will call people's attention to this opportunity.

Acknowledgements

We wish to thank the following developers for providing information about their systems: Kenneth M. Anderson, Hugh Davis, Kaj Grønbæk, Chuck Kacmar, Antoine Rizk, and Uffe Kock wiil. We gratefully acknowledge the support of 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 Jersery 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.

References

[1] K.M. Anderson, Integrating Open Hypermedia Systems with the World Wide Web, Proceedings of Hypertext'97, 1997, pp. 157–166
[2] M. Bieber, Automating hypermedia for decision support, Hypermedia 4 2 (1992) 83–110.
[3] M. Bieber, C. Kacmar, Designing hypertext support for computation applications, Communications of the ACM 38 8 (1995) 99–107.
[4] M. Bieber, H. Oinas-Kukkonen, V. Balasubramanian, Hypertext functionality, ACM Computing Surveys (forthcoming).
[5] M. Vitali, F. Ashman, H. Balasubramanian, V. Oinas-Kukkonen, Fourth generation hypermedia: some missing links for the world wide web, International Journal of Human Computer Studies 47 (1997) 31–65 (Available at: http://www.hbuk.co.uk/ap/ijhcs/websability/).
[6] J. XML Bosak, Java, and the future of the Web, Available at: http://sunsite.unc.edu/pub/sun-info/standards/xml/why/xmlapps.htm, 1997.
[7] T. Bray, J. Paoli, C.M. Sperberg-McQueen, Extensible Markup Language (XML). 1.0, Available at: http://www/w3.org/TR/1998/REC-xml-19980210, 1998.
[8] L. Carr, D. De Roure, W. Hall, G. Hill, The Distributed Link Service: A Tool for Publishers, Authors and Readers, In Proceedings of the Fourth International www Conference, 1995, pp. 647–656.
[9] D. Connolly, R. Khare, 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.
[10] H. Davis, W. Hall, I. Heath, G. Hill, R. Wilkins, Towards An Integrated Information Environment With Open Hypermedia Systems, ECHT'92 Proceedings, 1992, pp. 181–190
[11] H. Davis, S. Knight, W. Hall, Light Hypermedia Link Services: A Study of Third Party Application Integrating, Proceedings of ECHT'94, 1994, pp. 158–166.
[12] H. Davis, A. Lewis, OHP: A Draft Proposal for a Standard Open Hypermedia Protocol, Proceedings of the Second Workshop on Open Hypermedia Systems, 1996.
[13] K. Grønbæk, N.O. Bouvin, L. Sloth, Designing Dexter-based hypermedia services for the World Wide Web, Hypertext'97 Proceedings, 1997, pp. 146–156.
[14] R. Khare, A. Rifkin, X marks the spot: using XML to automate the web, IEEE Internet Computing 1 (1997) 78–87.
[15] K.C. Malcolm, S.E. Poltrock, D. Schuler, Industrial Strength Hypermedia: Requirements for a Large Engineering Enterprise, Hypertext'91 Proceedings, 1991, pp. 13–24.
[16] P.J. Nürnberg, J.J. Leggett, E.R. Schneider, J.L. Schnase, Hypermedia Operating Systems: A New Paradigm for Computing, Hypertext'96, 1996, pp. 194–202.
[17] K. Østerbye, U.K. Will, The Flag Taxonomy of Open Hypermedia System, Hypertext'96, 1996, pp. 129–139.
[18] J.L. Schnase, J.J. Leggett, D.L. Hicks, NüP.J. rnberg, J.A. Schez, Open Architectures for Integrated Hypermedia-based Information System, Proceedings of the 27th Hawaii International Conference on System Sciences, 1994, pp. 386–395.
[19] J. Wan, Integrating Hypertext into Information Systems through Dynamic linking, PhD dissertation, New Jersey Institution of Technology, Newark, NJ 07102, 1996.
[20] E.J. Whitehead, An Architectural Model for Application Integration in Open Hypermedia Environments, Hypertext'97, 1997, pp. 1–12.
[21] U.K. Wiil, J.J. Leggett, The HyperDisco Approach to Open Hypermedia Systems, Hypertext'96, 1996, pp. 140–148.
[22] U.K. Wiil, J.J. Leggett, Workspace: The HyperDisco Approach to Internet Distribution, Hypertext'97, 1997, pp. 13–23.