Spring 5-31-2001

Augmenting applications with hyper media, functionality and meta-information

Roberto Galnares

New Jersey Institute of Technology

ABSTRACT

AUGMENTING APPLICATIONS WITH HYPERMEDIA FUNCTIONALITY AND META-INFORMATION

by

Roberto Galnares

The Dynamic Hypermedia Engine (DHE) enhances analytical applications by adding relationships, semantics and other metadata to the application's output and user interface. DHE also provides additional hypermedia navigational, structural and annotation functionality. These features allow application developers and users to add guided tours, personal links and sharable annotations, among other features, into applications. DHE runs as a middleware between the application user interface and its business logic and processes, in a n-tier architecture, supporting the extra functionalities without altering the original systems by means of application wrappers.

DHE automatically generates links at run-time for each of those elements having relationships and metadata. Such elements are previously identified using a Relation-Navigation Analysis. DHE also constructs more sophisticated navigation techniques not often found on the Web on top of these links. The metadata, links, navigation and annotation features supplement the application's primary functionality.

This research identifies element types, or “classes”, in the application displays. A "mapping rule" encodes each relationship found between two elements of interest at the "class level". When the user selects a particular element, DHE instantiates the commands included in the rules with the actual instance selected and sends them to the appropriate destination system, which then dynamically generates the resulting *virtual* (i.e. not previously stored) page. DHE executes concurrently with these applications, providing automated link generation and other hypermedia functionality. DHE uses the eXtensible Markup Language (XML) - and related World Wide Web Consortium (W3C) sets of XML recommendations, like Xlink, XML Schema, and RDF - to encode the semantic information required for the operation of the extra hypermedia features, and for the transmission of messages between the engine modules and applications.

DHE is the only approach we know that provides automated linking and metadata services in a generic manner, based on the application semantics, without altering the applications. DHE will also work with non-Web systems.

The results of this work could also be extended to other research areas, such as link ranking and filtering, automatic link generation as the result of a search query, metadata collection and support, virtual document management, hypermedia functionality on the Web, adaptive and collaborative hypermedia, web engineering, and the semantic Web.

AUGMENTING APPLICATIONS WITH HYPERMEDIA FUNCTIONALITY AND META-INFORMATION

by
Roberto Galnares

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

Department of Computer & Information Sciences

May 2001

APPROVAL PAGE

Augmenting Applications with Hypermedia Functionality and Meta-Information

Roberto Galnares

Michael Bieber, Ph.D., Thesis Adviser — Date
Associate Professor of Information Systems, NJIT

Murray Turoff, Ph.D., Committee Member — Date
Chairman and Distinguished Professor of Information Systems, NJIT

Vincent Oria, Ph.D., Committee Member — Date
Assistant Professor of Information Systems, NJIT

Ravi Paul, Ph.D., Committee Member — Date
Assistant Professor of Information Systems, NJIT

Vassilka Kirova, Ph.D., Committee Member — Date
Software Technology Center, Bell Labs, Lucent Technologies

BIOGRAPHICAL SKETCH

Author: Roberto Galnares

Degree: Doctor of Philosophy in Computer and Information Science

Date: May 2001

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer and Information Science
  New Jersey Institute of Technology, Newark, NJ, 2001

- Ingeniero en Cibernética y Ciencias de la Computación
  Universidad la Salle, C.V. Mexico City, Mexico, 1984

Major: Computer and Information Sciences

Presentations and Publications:

Anirban Bhaumik, Deepti Dixit, Roberto Galnares, Manolis Tzagarakis, Michalis Vaitis, Michael Bieber, Vincent Oria, Qiang Lu, Firas Aljallad and Li Zhang,
Integrating Hypermedia Functionality into Database Applications,
Developing Quality Complex Database Systems: Practices, Techniques and Technologies, Shirley Becker (ed.), forthcoming.

Anirban Bhaumik, Deepti Dixit, Roberto Galnares, Manolis Tzagarakis, Michalis Vaitis, Michael Bieber, Vincent Oria, Qiang Lu, Firas Aljallad and Li Zhang,
Towards Hypermedia Support for Database Systems,
Proceedings of the 34th Hawaii International Conference on System Sciences, IEEE Press, Washington, D.C., January 2001.

Michael Bieber, Roberto Galnares and Qiang Lu,
Web Engineering and Flexible Hypermedia,
2nd Workshop on Adaptive Hypertext and Hypermedia, Hypertext '98 Conference, Pittsburgh, 1998.

Michael Bieber and Roberto Galnares,
Automated Hypermedia Support for the Virtual Documents Generated by Analytical Applications,
Workshop on Virtual Documents, Hypertext Functionality and the Web, WWW8 Conference, Gold Coast, Australia, 1998.

To my family

ACKNOWLEDGMENT

I would like to express my deepest appreciation to Dr. Michael Bieber , who not only served as my research supervisor, providing valuable and countless resources, insight, and intuition, but also constantly gave me support, encouragement, and reassurance. Special thanks are given to Dr. Ravi Paul, Dr. Murray Turoff, Dr. Vassilka Kirova and Dr. Vincent Oria for participating in my committee.

I would also like to express my gratitude to the generous organizations that provided the financial support that made this studies possible: the Department of Computer and Information Science at the New Jersey Institute of Technology, the U.S. Fulbright Commission, the Consejo Nacional de Ciencia y Tecnología (CONACYT, Mexico), as well as the professors who provided me with funding and guidance: Dr. Michael Bieber, Dr. James Geller, Dr. Yehoshua Perl, Dr. James McHugh, and Dr. Joseph Leung.

Many of my fellow graduate students in the Collaborative Hypermedia Research Lab are deserving of recognition for their contributions to this work. I also would like to thank my wife, Ana L. Escalante, for her love and support over the years.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

LIST OF FIGURES

CHAPTER 1

OVERVIEW

1.1 Motivation

Most information systems, including legacy systems, are designed without taking into consideration the advantages that distributed relational environments and hypermedia could offer to their users. This situation arises in part because, currently, systems architects and analysts usually do not employ a hypermedia design methodology as part of the system design process, and also because manually incorporating hypermedia in documents and applications is an arduous, often overlooked, task.

One helpful feature in the process of making existing or new applications accessible through the World Wide Web (WWW), or even during the deployment of hypermedia enabled systems not in the web, would be the automatic generation and inclusion of links and other metadata. This is especially important for engineering, scientific and business applications that generate their displays and results dynamically, instead of retrieving pre-existing documents. In those cases links could not be added manually in time to support the user, who would benefit if links and metadata were available at the time the screen or document is displayed.

As more and more applications are brought onto the Web, developers are beginning to perceive the depth of functionality that could be provided by adding hypertext features such as non-linear structuring and linking, annotation, and sophisticated navigation. Yet most users and developers seem unaware that most Web environments give them only a subset of hypermedia functionality [Bieber 97b].

Additional structuring features include semantically typed anchors, links and nodes; attributes and keywords on these; links to and from content spans within any medium; bi-directional links; links recursively to links and to other hypermedia constructs; transclusions/inclusions providing access to all uses of the same span of content; versioning of hypermedia constructs; landmarks; automatic link propagation; trails and guided tours; and local and global overviews. Navigation features include link traversal; visual feedback upon link arrival; search based on the hypermedia structure (as opposed to content); process enactment through link traversal; backtracking strategies; back-jumping; and history mechanisms. Annotation features include comments, bookmarks and reader-authored links with private, workgroup and public access. People should be able to annotate any hypermedia construct, including other annotations recursively [Vitali 00].

Several new standard Web data formats and protocols can facilitate the implementation of many of these functionalities. Now that the basic hypermedia building blocks exist, hypermedia and www researchers must now figure out how to deploy them to create an infrastructure for seamlessly integrating hypermedia techniques into the Web environment and making them pervasive in everyone's daily Web activities [Vitali 00].

Hypermedia researchers believe strongly that readers should be able to author at will, with the ability to add links and annotations to any document, and this must take place in the browser as the user reads. Furthermore, these browsers should support the creation of hypertext constructs such as semantic node and link types and other attributes, trails and guided tours, etc. Also, browsers should automatically display destination link spans, and metainformation such as semantic types, keywords and other attributes as part of maintaining the user's orientation.

Hypermedia features such as bi-directional linking, personalized links and structural search rely on links as "first class objects" maintained external to documents instead of embedded inside anchors. Ubiquitous linking from documents not belonging to the user requires links and anchors be maintained external to documents and only merged as the document is sent for display. This requires external linkbases and anchor points maintained outside documents.

Unless an HTML document is dynamically generated, anyone without write-access can neither create anchors at a link destination within that document, nor create links leading from that document. Several interesting hypermedia features remain impossible until we can extract link anchors from document content, including bi-directional linking (links activated at both ends, and thus not providing a preferred direction of navigation) and personalized links (visible to a group of users, rather than all readers of a document).

Two issues arise when separating content and links. The first is the problem of locators, i.e., of unequivocally identifying the exact spot in the document at which link anchors should be attached. Of course, this is extremely media type-dependent; methods to specify a location within ASCII documents, marked-up texts or, say, bitmaps will vary considerably. The second issue is keeping the locators correctly pointing to the right positions when either or both linked documents are being modified asynchronously. These issues are even more complex in the case of virtual documents that are generated dynamically. Nowadays there is a myriad of applications which construct their displays automatically as result of input submitted by users. Virtual documents do not exist (in the form of a file) prior to generation, raising several potential problems regarding their management, like how to refer to them, and how to add hypermedia functionality to them.

1.2 Proposed Solution: the Dynamic Hypermedia Engine (DHE)

Most applications give users limited access to the relationships inherent with these applications and to relationships external to the application [Bieber 97a]. An application's knowledge base can be administered and organized to manage the interrelationships between the various data objects. These relationships render more contextual knowledge to users interacting with the application by providing direct access to information not immediately accessible and consequently often not fully understood.

There is a primary need for automating such functionality into applications just so that the interrelationships within the application's knowledge are made visible to the user. Furnishing these associations shows new ways to view the application knowledge, navigate among items of interest, and annotate comments and relationships. Thus, not only would the user benefit from the additional hypermedia information but also from being able to widen his mental model of the application domain. Because this kind of support is missing from most Web applications and from all standard Web environments, we believe it imperative to improve hypertext support for Web applications.

These problems are accentuated in the legacy applications [Bennett 95] increasingly being transferred to the Web in order to gain universal access (outside or even inside an organization) or to achieve platform independence. These applications are being converted with the minimal amount of reengineering, and rarely take advantage of the existence of links beyond providing a home page, index, table of contents, next page button, etc. [Bieber 97a]. Assisting and automating Web conversion could alleviate the aforementioned problems and actually enhance legacy applications to take advantage of the Web's hypermedia features.

The Dynamic Hypermedia Engine (DHE) enhances analytical applications dynamically, adding relationships, semantics and other metadata to the application's output and user interface. The DHE hypermedia engine executes as middleware between the target application and its user interface, providing additional hypermedia navigational, structural and annotation functionality.

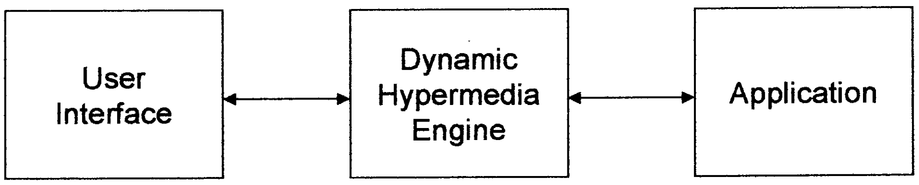


Figure 1.1 DHE as Middleware

'Wrappers' need to be written for each application in order to integrate them with the engine architecture. The use of application wrappers enables DHE to provide these extra functionalities without altering the original systems. Applications or their wrappers then connect to DHE using messages delivered through the network.

DHE 'intercepts' all messages passing between the application and the user interface, and uses the mapping rules specified above to map each appropriate element of the message to a hypermedia node or anchor. The user interface wrapper integrates these anchors into the document being displayed and sends it to the user's user interface. When the user selects an anchor, the browser wrapper passes it to DHE, which returns a list of possible links (one for each appropriate relationship as determined by the mapping rules).

If the user selects a hypermedia engine link (e.g., to add an annotation or stage in a guided tour), DHE processes it entirely. If the user selects a relationship with a destination in a registered application, DHE infers and instantiates the appropriate application commands from the relationship's mapping rules and passes them to the appropriate application for processing. If the user selects a user-created annotation or tour, etc., DHE retrieves it. Thus DHE automatically provides all hypermedia linking (as well as navigation) to applications, which remain hypermedia-unaware and in fact often entirely unchanged. DHE also constructs more sophisticated navigation techniques not often found on the Web (e.g., guided tours, overviews, structural query) on top of the dynamically generated links. The metadata, links and navigation, as well as annotation features, supplement the application's primary functionality.

1.3 Main Contributions

The Dynamic Hypermedia Engine is a modular distributed middleware able to enhance Information Systems with metadata, along with hypermedia structuring, navigation and annotative functionality.

DHE is the only approach we know that provides automated linking and metadata services in a generic manner, based on the application semantics (as opposed to search or lexical analysis), without altering applications. It is uniquely suited to support analytical and technical applications that generate the contents of their displays dynamically.

The DHE prototype offers a solution to the problem of how to include linking and metadata functionality to a virtual document. It will also become the framework and test bed on which further research could be accomplished, like the development of additional hypermedia functionality and the application of empirical experiments.

The updated DHE architecture will become the new test bed for the implementation of new hypermedia functionalities, and offers a solution for several of the problems related with the management of dynamically generated documents.

There are several areas in which this research could be applied and extended, such as: link ranking and filtering, automatic link generation as the result of a search query, metadata collection and support, virtual document management, hypermedia functionality on the Web, adaptive and collaborative hypermedia, and distributed applications architecture design.

1.4 Outline

Chapter 1, Overview, states what the problem is, describes the approach followed to solve it, and the goals and contributions of this research.

Chapter 2, Hypermedia, reviews the hypermedia field giving special attention to hypermedia functionality and support, as well as virtual documents and automatic link generation.

Chapter 3, Metadata, examines the concept and applications of metadata, and gives a general survey of the different metadata standards currently in use. The term 'Meta-Information' is introduced.

Chapter 4, Dynamic Hypermedia Engine, presents an overview of the Dynamic Hypermedia Engine (DHE), its main components, and the principles of its operation.

Chapter 5, DHE 1.0, describes the initial implementation of the Dynamic Hypermedia Engine, named DHE 1.0, it uses a detailed example to show the added functionalities, and offers a critique of its performance and usability.

Chapter 6, DHE NG, shows the requirements and conceptual architecture of the next version of the Dynamic Hypermedia Engine (DHE NG).

Chapter 7, Results and Future Research, summarizes the results produced by this research and moves ahead to propose additional, or extended, research topics.

Appendix A, DHE NG Data Structures, contains a formal documentation of some of the data structures required by the Next Generation of the Dynamic Hypermedia Engine.

1.5 Summary

Most information systems are designed without considering hypermedia. The dynamic generation of links and metadata could be helpful in the deployment of application on the WWW.

New standards facilitate the implementation of hypermedia functionalities. Researchers must find out how to integrate hypermedia techniques in the Web environment.

The Dynamic Hypermedia Engine (DHE) enhances analytical applications dynamically. DHE executes as middleware providing additional hypermedia functionality.

DHE's unique approach provides dynamic linking based on the application semantics. It supports applications that generate dynamic content.

DHE also serves as test bed for further research, and could be applied and extended in several areas.

CHAPTER 2

HYPERMEDIA

The Web owes its origins to many people, starting back in medieval times with the development of a rich system of cross references and marginalia. The basic document model for the Web was set: things in the page such as the text and graphics, and cross references to other works. These early hypertext links were able to able to target documents to a fine level thanks to conventions for numbering lines or verses.

Vannevar Bush in the 1940's, in his article ‘As we may think’ [Bush 45], describes his vision for a computer aided hypertext system he named the ‘memex’. His description of browsing a Web of linked information, includes the user’s ability to easily insert new information of their own, to add to the growing web. Dr. Bush was the Director of the US Office of Scientific Research and Development, and coordinated war time research in the application of science to war.

Other visionaries include Douglas Engelbart, who founded the Augmentation Research Center at the Stanford Research Institute (SRI) in 1963 [Engelbart 63]. He is widely credited with helping to develop the computer mouse, hypertext, groupware and many other seminal technologies. In 1968, Engelbart introduced NLS, the *oN Line System*, as an experimental tool to store all specifications, plans, designs, programs, documentation, reports, etc. Throughout the years the system grew to over 100,000 information items. The computer terminals were very sophisticated, including video projectors, special keyboards, and mice. He now directs the Bootstrap Institute, which is dedicated to the development of collective IQ in networked communities.

Ted Nelson has spent his life promoting a global hypertext system called Xanadu. He coined the term hypertext [Nelson 65], and is well known for his books: Literary Machines [Nelson 81] and Dream Machines [Nelson 74], which describe hypermedia; including branching movies, such as the film at the Czechoslovakian Pavilion at Expo `67.

One of the first hypertext-based systems was developed in 1967 by a team of researchers led by Dr. Andries van Dam at Brown University. The research was funded by IBM and the first hypertext implementation, Hypertext Editing System, ran on an IBM/360 mainframe. IBM later sold the system to the Houston Manned Spacecraft Center which reportedly used it for the Apollo space program documentation. A year later, in 1968, van Dam developed FRESS, a File Retrieval and Editing System which was an improvement of his original Hypertext Editing System and was used commercially by Philips.

In 1972, researchers at Carnegie-Mellon University began development of ZOG (doesn't stand for anything). ZOG was a large database designed for a multi-user environment. The ZOG database consisted of frames which, in turn, consisted of a title, a description, a line with standard ZOG commands, and a set of menu items (called selections) leading to other frames. The ZOG database was text-only and originally ran on an IBM mainframe. A PERQ workstation implementation of ZOG was used on the nuclear-powered aircraft carrier USS Carl Vinson. Two of the original developers of ZOG, Donald McCracken and

Robert Akscyn, later developed KMS, Knowledge Management System, which was an improved version of ZOG. KMS runs on Sun and HP Apollo workstations with much enhanced performance. Though KMS included a GUI, it still remained a text-based system. It was intended to be a collaborative tool, in that users could modify the contents of a frame and the changes would be immediately visible to others through dynamically updated links. [DeBra 98]

In 1978, Andrew Lippman of MIT Architecture Machine Group, lead a team of researchers that developed what is argued to be the first true hypermedia system called the Aspen Movie Map. This application was a virtual ride simulation through the city of Aspen, Colorado. Four cameras, pointing in different directions, were mounted on a truck which was driven through the streets of Aspen. The cameras took pictures at regular intervals, and all the pictures were compiled onto videodisks. The images were linked in such a way that would allow the user to start at a given point and move forward, back, left, or right. Once a route through the city was chosen, the system could display the images in rapid succession creating a movie-like motion. The system also included images of the interior of several landmark Aspen buildings, so the user could take a virtual tour of these buildings. Another interesting feature of the system was a navigation map which was displayed in addition to the movie window. The user could jump directly to a point on the city map instead of finding the way through the city streets to that destination. The Aspen Movie Map was a landmark in hypermedia development in that, through a sophisticated application, it demonstrated what could be achieved with the technology available at the time [DeBra 98].

Bill Atkinson best known for MacPaint, the first bitmap painting program, gave the world its first popular hypertext system HyperCard [HyperCard]. Released in 1987, HyperCard made it easy for anyone to create graphical hypertext applications. It features bitmapped graphics, form fields, scripting, fast full text search, manual linking, and backtracking. HyperCard is based on a stack of cards metaphor with shared backgrounds. HyperCard was first used as a tool for building hypertext systems with, instead of being considered a hypertext system by itself [Nielsen 90]. It spawned imitators such as the Asymmetrix Toolbook which used drawn graphics and ran on the PC. The OWL Guide was the first professional hypertext system for large scale applications, it predates HyperCard by one year and followed in the footsteps made by Xerox NoteCards, a Lisp-based hypertext system, released in 1985. Jeff Conklin describes several other early hypertext systems [Conklin 87].

The ACM SIGWEB [SIGWEB], formerly SIGLINK, has for many years been the center for academic research into hypertext systems, sponsoring a series of annual conferences. SIGLINK was formed in 1989 following a workshop on hypertext, held in 1987 in Chapel Hill, North Carolina. Tim Berners-Lee and Robert Caillau both worked at CERN, an international high energy physics research center near Geneva. In 1989 they collaborated on ideas for a linked information system that would be accessible across the wide range of different computer systems in use at CERN. At that time many people were using TeX and Postscript for their documents. A few were using SGML. Berners-Lee [Berners-Lee 89] realized that something simpler was needed that would cope with dumb terminals through high end graphical X Windows workstations. HTML was conceived as a very simple solution, and matched with a very simple network protocol HTTP.

CERN launched the Web in 1991 along with a mailing list called www-talk. Other people thinking along the same lines soon joined and helped to grow the web by setting up Web sites and implementing browsers, such as, Cello, Viola, and MidasWWW. The break through came when the National Center for Supercomputer Applications (NCSA) at Urbana-Champaign encouraged Marc Andreessen and Eric Bina to develop the X Windows Mosaic browser. In order to speed development, Mosaic dropped many of the features Tim Berners-Lee originally envisioned and included in his first prototype, such as user editing. Mosaic was later ported to PCs and Macs and became a run-away success story. The Web grew exponentially, eclipsing other Internet-based information systems such as WAIS, Hytelnet, Gopher, and UseNet.

All these systems that supplied linking, required the author to add the links manually (with some exceptions in Electronic Commerce systems that put single-destination links on database retrievals, and systems that generated links from keyword searches).

2.1 Hypermedia Functionality

Until recently, few system developers actively thought about an application's interrelationships, and whether users should access and navigate along these relationships directly [Ashman 99]. The Hypertext Functionality (HTF) field studies techniques for and the impact of supplementing everyday computer applications with hypertext functionality. Hypertext structuring, annotation and navigational functionality can enrich business, scientific, engineering and personal applications, thereby making them more effective [Oinas-Kukkonen 95].

People use these applications primarily for their underlying analytical functionality, i.e., not for their ability to produce hypertext documents or displays. It is unlikely that users of such applications will change in favor of other applications just because they offer hypertext. Developers therefore must find it relatively easy to retrofit HTF to existing applications, as well as incorporate HTF into new ones. HTF should be integrated and deployed so seamlessly that users do not find its presence at all out-of-place [Bieber 00]. Augmenting applications with direct access and hypertext structuring, annotation and navigation functionality [Bieber 97a] 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.

Hypertext Structuring Functionality: Hypertext structuring functionality includes local and global information overviews; alternate views and contexts; transclusions that preserve context by "including" the original content at all places that use it and maintaining links between all these uses; link propagation; node, link and anchor typing; as well as keywords and other attributes on all of these [Bieber 00].

Hypertext Navigation Functionality: Navigation functionality encompasses access ranging from information retrieval to browsing. This includes content- and structure-based query; history-based navigation and sophisticated backtracking; bi-directional linking; dynamic and computed linking; and process enactment or execution through link traversal [Bieber 00].

Hypertext Annotation Functionality: Annotation includes bookmarks, landmarks, manual linking and commenting. Note that many of these features can be shared in collaborative environments. Many also can be personalized for different users and tasks [Bieber 00].

The Web finally provides a platform for widespread HTF support. Few Web applications (and even fewer off the Web), currently take more than modest advantage of hypertext [Bieber 97a], [Bieber 97b]. Thus, HTF researchers have the opportunity to make a major impact on how applications of the future will look, and on the level of support and quality of interaction that users will come to expect.

Many of the ideas presented here came from shorter papers presented at various Hypertext Functionality Workshops. Since 1994, the HTF research community has organized many workshops [HTF 94], [HTF 96], [HTF 97], [HTF 98a], [HTF 98b], [HTF 98c] and more are scheduled. HTF research closely parallels research in organizational hypermedia - hypertext support of the information systems field. Since 1993 the primary forum for this research has been the Hypermedia in Information Systems and Organizations mini-track at the

Hawaii International Conference on Systems Sciences [HICSS 93]. A special journal issue also resulted from this forum [JOC 96]. In 1998 this mini-track was expanded into the Web Information Systems mini-track [HICSS 98], recognizing the Web as today's primary platform for hypertext development. There is also a special issue on hypertext functionality in the Journal of Digital Information [JoDI 99].

2.2 Hypermedia Support

Hypermedia data models

Several hypermedia data models, such as the Hypertext Abstract Machine (HAM) [Campbell 88], Dexter [Halasz 94] and the World Wide Web's HTML provide data structures and guidelines for constructing nodes, links and anchors, and for browsing. Builders can implement one of these instead of making up their own. HyTime [HyTime] provides an SGML-based ISO standard model for hypermedia documents (as well as musical documents), although its complexity has discouraged all but a few implementations. Implementing a data model is no easy task, but it does provide a solid starting point for a hypermedia system.

The Toolkit approach

The toolkit approach facilitates incorporating hypermedia functionality into individual non-hypermedia applications being built from scratch. An application builder embeds calls to the toolkit's subroutine library in the application's code. Examples of toolkits include the Hypertext Object-oriented Toolkit [Puttress 90] and the Andrew Toolkit [Sherman 90]. Anderson [Anderson 96] extended the JAVA ATW user-interface toolkit with hypermedia-aware widgets. Garrido and Rossi [Garrido96] extended the VisualWorks Smalltalk palette to include hypermedia-aware widgets.

Hyperbases

Builders constructing hypermedia systems from scratch need to store and retrieve hypermedia constructs (nodes, links and anchors), which persist between user sessions. Hyperbase storage engines such as HB3 [Leggett 94] provide a data store and management routines for hypermedia constructs akin to database management systems. System builders can integrate hyperbases into their systems, as many link services and open hypermedia systems do.

Link Services

Several research efforts have succeeded in creating links among primarily non-hypermedia information systems and in facilitating their traversal. Microcosm [Davis 94], Multicard [Rizk 92], and Chimera [Anderson 97] each has a hypermedia link service that executes concurrently with external systems and provides linking support. The Distributed Link Service consisted of an unenclosed environment providing a navigational overlay to Web pages based on within-node text analysis to identify implicit link opportunities such as key phrases, personal names and bibliographic citations [Carr 98].

Each of those hypermedia systems expects client information systems to support anchor creation and selection by embedding hypermedia calls and handling minimal hypermedia functionality in a manner similar to the toolkit approach. They also provide multiple levels of hypermedia navigation and annotation support, based on the degree of hypermedia compliance the client non-hypermedia information system provides, as well as a limited set of support for client systems that are not hypermedia compatible at all. All systems, however, concentrate primarily on supporting manual linking within display-oriented systems.

Generating hypermedia applications from a hypermedia design methodology

Hypermedia design methods present one of the most important developments in the hypermedia field [Yoo 00], [Bieber 95a]. This involves more than applying well-understood system analysis and design or software engineering techniques to hypermedia. Hypermedia functionality requires new kinds of relationship management and navigation support. Hypermedia design methods provide a systematic approach to relationship management and navigation support, which in turn, should enable consistent, large-scale, robust hypermedia implementations. Most of the design methods available produce new hypermedia applications, complete with their own interface. Our approach differs in that it supports new or existing applications, without requiring hypermedia-oriented relationships or features to be embedded in the application code.

Hypermedia application software

For standalone applications, many commercial and research hypermedia application environments exist. Here are just a few. The MacWeb design environment produces a hypermedia application for knowledge-based systems [Nanard 95]. VIKI enables users to build spatial hypermedia applications in which all links are implicit through the user's spatial location of nodes [Marshall 94]. These often support only manual hypermedia linking, though some may be extended through a scripting language.

Open Hypermedia Systems

Open hypermedia research addresses the issues of integrating hypermedia functionality into existing applications in the computing environment. An open hypermedia system (OHS) is typically a middleware component in the computing environment offering hypermedia functionality to applications orthogonal to their storage and display functionality. Using the services of an OHS, existing applications in the computing environment can become "hypermedia enabled", thus supporting linking to and from information managed by the application without altering the information itself. To become "hypermedia enabled", applications must be extended to make the hypermedia functionality available in the user interface and must be able to communicate hypermedia requests to the OHS. The term open hypermedia environment is used to cover both the OHS and the set of hypermedia enabled applications. An open hypermedia environment is a subset of the overall computing environment in terms of applications, programs and services.

Open Hypermedia Systems are currently being deployed in various application domains, including digital libraries, computing support for large engineering enterprises, software development and education. Yet, open hypermedia developers have until recently lacked consensus, common guidelines and standards for interoperation between OHSs and applications that request hypermedia services [Nürnberg 98]. Without such guidelines and standards, each open hypermedia system uses different approaches and techniques that inhibit interoperability. The immediate results of this are:

- *Lack of presentation interoperability*: applications enabled for one particular OHS cannot be used with other OHSs. Thus, currently, each application has to be enabled separately for each OHS instead of just once for all OHSs.
- *Lack of storage interoperability*: hypermedia structures built using one particular open hypermedia environment cannot be accessed and used by other open hypermedia environments. Thus, currently, each open hypermedia environment is an "island" instead of a part of a larger, unified hypermedia environment.
- *Lack of system interoperability*: hypermedia structures cannot be built across different open hypermedia environments. It is, for instance, not possible to create a link with one end in one particular open hypermedia environment and another end in another open hypermedia environment. This adds to the

problem of open hypermedia environments being "islands" as mentioned above.

The open hypermedia community is currently addressing interoperability issues by defining a set of guidelines and standards for interoperation. A working group [OHSWG] has been formed, and a number of meetings have been held in this ongoing standardization effort.

2.2.1 Hypermedia Engines

Hypermedia engines execute independently of an application with minimal modifications to it, and provide the application's users with hypermedia support. Few approaches provide transparent hypermedia integration as DHE does. Notable projects include Microcosm's Universal Viewer [Davis 94], Freckles [Kacmar 95], the OO-Navigator [Garrido 96], and NBCi's QuickLink [QuickLink]. The Universal Viewer gives users access to a minimal level of Microcosm's hypermedia functionality through the PC Windows operating system. It places selectable buttons in the title bar of the application's windows. Users can select text in these unaware applications and select a Microcosm button to check for "generic" or "keyword" links originating from any portion of the text selected. Links to these applications will launch the application under the Universal Viewer and open the destination document within it.

Kacmar's [Kacmar 93] work focuses on hypermedia-aware interfaces. His engine works together with the interface to determine how to display objects. The hypermedia engine handles linking and traversal. The interface allows users to select anchors and display the results of traversals. The underlying back-end application must be implemented so that it returns an object's contents when requested. Kacmar's work differs from DHE, in part, through its philosophy. His work aims at generating hypermedia-aware interfaces, where our work eventually should lead to hypermedia-unaware interfaces as well as hypermedia-unaware applications.

Kacmar's second project [Kacmar 95] developed another version of his architecture which operates entirely independently of the interface system. It uses the underlying UNIX X Windows system to overlay all hypermedia controls (nodes, anchors, menus, etc.). As with the Universal Viewer, it remains unaware of the application's data model and object identifiers. It uses the window title, as a structured object descriptor, to associate an identifier with the window's contents. Window contents are maintained by the underlying application.

All three approaches seamlessly support an application's other functionality but provide only manual linking - in the Universal Viewer's case on an object's display value; in Kacmar's first system on the object identity; and in Kacmar's second system, on the object's fixed location within a recognized window. DHE, in contrast, contributes by providing seamless, automated supplemental navigation and interrelationship access to dynamically-mapped information systems.

The OO-Navigator comes the closest to our approach, providing a seamless hypermedia support for computational systems that execute within a single Smalltalk environment. They provide a hypermedia layer above the applications which maps application classes to node classes, and relationships among classes to link classes. From these mappings the OO-Navigator automatically generates nodes and links. As mentioned earlier, the OO-Navigator uses Smalltalk's interface modified with hypermedia-aware widgets [Garrido 96]. This approach meets DHE's goal of supplementing Smalltalk applications with hypermedia support without altering them. DHE's approach applies to both object-oriented and non object-oriented applications.

QuickClick [QuickClick] is a recent application that allows users to access pop-up lists of links related to a particular word. When the user activates QuickClick, either by clicking a yellow underline or alt-clicking any word, a window pops up containing the list of links with additional information about that word. QuickClick allow users to define their own links through BoosterPacks. BoosterPacks are collections of keywords and destinations that are created by users. QuickClick offers some of the functionality DHE provides, but it does so based on lexical values (i.e. words). DHE works mainly on semantic types.

2.3 Virtual Documents

A virtual document is a document for which no persistent state exists and for which some or all of each instance is generated at run time [Watters, 1999]. The paradigm of the Web has shifted the expectations for access to information. Previously, information was accessed by the retrieval of electronic copies of documents from a large repository of relatively static information. Now information is accessed through the manipulation of a large collection of information resources. Some of these resources are documents and some of these resources are processes that create documents.

A number of interesting research issues must be resolved surrounding these virtual documents on the Web:

- *Generation* - At what point in time is a virtual document defined? A virtual document can be defined by an author through the use of templates and links or it can be defined as the result of a search or application. Guided tours can be generated dynamically, based on an information need as defined by a user profile and/or an explicitly stated query.
- *Search* - How do you search for virtual documents? What is the domain in which to perform the search? Will the document exist by the time the user requests it?
- *Revisiting* - Users have an expectation that documents found once will be available on a subsequent search. The notion of bookmark does not apply to virtual documents in its normal, simplistic way. Bookmarks need enough information to recreate the document as it was.
- *Versioning* - Version control has long been a concern of Information Retrieval research and is now a central issue for management of virtual documents. Users need to be able to return to a bookmarked version of a virtual document and to go forward and backward in time through changes to that virtual document.

- *Authentication* - Who is responsible for the quality of the contents of a virtual document where components may come from a variety of sources and /or processes?
- *Reference* - How do authors cite virtual documents or versions of virtual documents?
- *Annotation* - The roles of user of information and supplier of information are merging. Readers expect to be able to add data, such as, comments, annotations, paths, and links, as well as content, while they are reading.
- *Hypertext Functionality* - How to provide hypertext functionality to virtual documents?

2.4 Automatic Link Generation

Traditional approaches to hypertext assumed that authors created well-defined (pre-defined) paths or trails in hypertext. When the reader approached the hypertext, the trail was already available. In this approach, inter-node links changed only when an author either updated or deleted them. The role of the author was to create the hypertext and the role of the user/reader was to browse through it. Thus, the reader was faced with the task of understanding the author's mental model of the hypertext documents in order to navigate the collection of linked nodes (hyperbase) effectively. There was no opportunity for readers to personalize the hypertext according to their interests, or for the types of links that were available to be adapted to the particular situation or task.

In some hypertext design models [Garzotto 1991] links do not necessarily exist explicitly and are implicit in the structure. In some cases they may be dynamically created by a process which defines how elements of the hypertext structure are related when the process is invoked [Nürnberg 97], [Marshall 97], [Wiil 99], [Shipman 99]. In open hypermedia systems [Davis 98] links may be manually or system created, are typically stored in a database independently of the documents, and can be static or dynamic.

One approach to dynamism in hypertext focuses on computing links based on relationships or similarities between texts or passages of text. In this approach, the link is not defined as a pointer from one hypertext node to another, but rather as a query that leads to a different node. Different forms of link types can then be envisioned based on how and when a query is constructed. Pre-computed links can be constructed at any time, whereas dynamic links are computed at the moment they are required [Ashman 97]. An example of a precomputed link would be a link from an employee's name occurring anywhere in a hyperbase to the biographical details for that person contained in the organization files. The advantage of such a precomputed link is that it could regulate all such instances of employee's names, and it could be updated automatically as the composition of the company changed. In contrast, a dynamic query (computed at run-time) can take into account the specifics of the current user interaction. The query might be based on a combination of the browsing history, user profile, content of the current document, etc. In this approach, dynamic hypertext represents an intermediate point on a continuum between querying and browsing [Waterworth 91].

In one form of query-based dynamic linking, queries are marked up directly on texts while they are being read [Golovchinsky 93]. For instance, clicking on the word "hypertext" and then dragging with the mouse button down to the word "search" (at which the point the mouse button is raised), would form the query "hypertext AND search". This form of query markup has been shown to yield reasonable results in large-scale text retrieval tasks [Charoenkitkarn 95]. The querying interface can be further modified so that users simply click on previously marked up text or phrases (e.g., marked up in blue with underlines, to imply the presence of a hypertext link). The text markup is created based on terms that have been previously selected, content-bearing words related to those terms, and possibly automated indexing techniques, including phrase identification methods. The sentence that the link (i.e., the marked up text that was clicked on) occurs in is sent to a search engine as a query. It is assumed that the user selects a particular link due to an interest in the content surrounding the link. The most relevant node to the query becomes the endpoint for the selected link. This creates a hypertext "point and click" interface to an underlying search functionality.

Query-based dynamic hypertext can look and feel like a static hypertext system. Interaction consists of clicking on marked up words or phrases in the text, which are then treated as if they were links. In experimental testing of this type of system [Tam 1997], users perceived the system to be a human-authored static hypertext, in spite of the fact that it was actually working as a search system with a highly simplified user interface. [Bodner 97] further describes this query-based model of dynamic hypertext.

Arguments in favor of query-based dynamic hypertext include: a simplified interface to search functionality, reduced authoring effort, and greater opportunity for customization based on the current user's interaction history or specific task context. An additional benefit is decreased cost of maintenance, since dynamic links do not get broken, and new material is automatically available for dynamic linking (i.e., automated updating of the implied hypertext structure is a side effect of dynamic linking). However, disadvantages include computation of each link at run-time (instead of using stored, precomputed links), which can be expensive in a large system with many users [Ashman 1997]; and over-completeness, which occurs when the reader is presented with more links then he/she can comprehend.

A combination between retrieval and navigation methodologies can result in the dynamic provision of links created on the fly using 'theming' technology based on statistical and linguistic techniques to provide links derived from the current document context. This technology has been successfully employed commercially [Multicosm] as an evolution of a dynamic link service [Carr 1999]. Content-based navigation (CBN) is embodied in the concept of the generic link [Hall 96]. The source anchors for such links are specified in terms of source content rather than source location. The storage of source content as part of the link structure was facilitated by the adoption of external link databases. Once authored from some source selection, a generic link may be followed from any matching instance of the source content: hence content-based navigation. The use of CBN gives substantial savings in authoring and link maintenance effort [Lewis 99]. One of the problems with using content, whether it is for retrieval or for navigation, is that it is not in the content that we are really interested. Words and pictures, videos and speech are representations of objects, ideas and concepts in the real world. This means dealing with the difference between the signifier (media representations) and the signified (real objects, concepts, ideas). It is the signified in which we are really interested [Smoliar 1996].

Humans can make the link from signifier to signified almost automatically, typically drawing on a huge body of prior knowledge. But in software systems the link (or its absence) is at the root of many of the problems with content-based retrieval and navigation [Grønbæk 96], [Davis 1999]. The same concept can have many different text representations even in the same language (synonyms). Furthermore, the same concept (e.g., car brand) can have many different instances (e.g., Mercedes-Benz, Ferrari, Porshe, etc.).

In an attempt to overcome some of the problems with text, the use of digital thesaurus tools or facilities for statistically based associations have been incorporated into information retrieval systems. More recently researchers have attempted to introduce layers of associations, above the media based links, which try to capture semantic associations relevant to an application and provide navigation and retrieval based on concepts in addition to content [Tudhope 99], [Cunliffe 97], [Beynon-Davies 94], [Bullock 98], [Nanard 91]. The MAVIS-2 project

[Dobie 99] and the COIR project [Hirata 96] both attempt to associate media based representations directly with a semantic layer in an attempt to provide integrated approaches to content and concept based retrieval and navigation.

Building semantic layers and associating media content with the concepts they represent is currently a labor intensive task. One of the challenges now is to build systems which extract or learn the semantics from the knowledge implicit in the media and make the associations between the media representations and the semantics without a heavy manual input [Lewis 99].

Today, almost every institution has its home Web site, with links connecting internal pages together. With huge amounts of information to be displayed and maintained, manually authoring and maintaining institution Web sites is almost infeasible. The Web pages of many of large sites are generated by programs, and database systems are used as backend data providers. A second generation of Web query languages and systems, defined by Florescu et al. [Florescu 98], extends the first generation by providing data organization, link generation, and document/structure authoring capabilities. Representative systems include Strudel [Fernandez 98] and WebOQL [Arocena 98].

2.5 Summary

Hypermedia and the Web owe their origins to many people: Vannevar Bush, Douglas Engelbart, and Ted Nelson conceived complex systems in which the information is inter-related and linked.

Tim Berners-Lee realized that something simpler was needed. HTML was conceived as a very simple solution, and was matched with a very simple network protocol HTTP. The NCSA encouraged the development of the Mosaic browser. In order to speed development, Mosaic dropped many of the features Berners-Lee originally envisioned and included in his first prototype. Web applications currently take limitted advantage of hypertext.

Hypertext Functionality provides techniques for supplementing everyday computer applications with hypertext. Hypertext structuring, annotation and navigational functionality can enhance applications.

Several hypermedia data models, such as the Hypertext Abstract Machine, Dexter, and HTML provide data structures and guidelines for constructing nodes, links, anchors, and for browsing.

Most systems that allow linking, require the author to add the links manually. There exist several approaches to facilitate incorporating hypermedia functionality into individual non-hypermedia applications, such as: toolkit, hyperbases, and Link Services.

Hypermedia design methods are one of the most important developments in the hypermedia field. They provide a systematic approach to relationship management and navigation support.

Open hypermedia research addresses the issues of integrating hypermedia functionality into existing applications. The open hypermedia community is currently addressing interoperability issues by defining a set of guidelines and standards for interoperation.

Hypermedia engines execute independently of an application with minimal modifications to it, and provide the application's users with hypermedia support.

A virtual document is a document for which no persistent state exists and for which some or all of each instance is generated at run time. A number of interesting research issues include: generation, search, revisiting, versioning, authentication, reference, annotation, and hypertext functionality

Traditional approaches to hypertext assumed that authors created well-defined (pre-defined) paths or trails in hypertext. The role of the author was to create the hypertext and the role of the user/reader was to browse through it. In some hypertext design models links do not necessarily exist explicitly and are implicit in the structure. In some cases they may be dynamically created by a process which defines how elements of the hypertext structure are related when the process is invoked.

Arguments in favor of dynamic hypertext include: a simplified interface, reduced authoring effort, greater opportunity for customization, and decreased cost of maintenance. Disadvantages include computation of each link, and over-completeness.

Building semantic layers and associating media content with the concepts they represent is currently a labor intensive task.

DHE contributes by providing seamless, automated supplemental navigation and interrelationship access to dynamically-mapped information systems.

CHAPTER 3

METADATA

Metadata is structured data that describes data. Increasingly large amounts of scientific and technical data are being created and saved in digital data storage systems. There is a need to expedite the access and use of this data. A variety of different data and formats need to be addressed, such as: images, video, audio, tables, arrays, graphics, algorithms and procedures, and documents. The term is frequently being employed to refer to any data used to aid the identification, description and location of networked electronic resources.

Metadata is not a concept exclusive to computer scientists but one that is used in other fields as well. Currently, computer science lacks a standard metadata framework broad enough to describe any kind of resource. The W3C metadata initiative [W3C Meta] is a movement to design comprehensive and collaborative systems for describing data.

DHE makes extensive use of metadata. The automatic link generation aspect of the DHE relies on the inclusion of semantic metadata for each pre-defined element present in a virtual document. Furthermore, DHE not only adds links to the displayed documents, it incorporates as much information about such elements as possible. In fact, DHE looks at links as just a particular kind of metadata, one that defines relationships between elements, objects and components. The term *MetaInformation* is used in this research to make clear when the metadata includes relationships.

This section explores some of the metadata proposals and recommendations that would be part of the process of designing a standard metadata framework. Such a framework will provide a standard way of describing resources and their relationships.

Resources are required to evolve the Web from their current state to one where they are organized, catalogued and effectively searchable; becoming what Tim Berners-Lee calls the 'Semantic Web' [Berners-Lee 99]. The proposals employ the extensible markup language for their syntax and model. XML, being extensible, allows the proposals to be extensible, making for complex and effective description. Metadata applications can be customized to user needs, without losing interoperability with other applications, allowing for accurate representations of resource relationships.

The Semantic Web Activity [W3C SW] has been recently established to lead both the design of enabling specifications and the open, collaborative development of technologies that support the automation, integration and reuse of data across various applications. The Semantic Web Activity builds upon the existing foundation work accomplished by the W3C Metadata Activity.

3.1 Introduction

Documents used to have only one tier of data; they were whole, internally formatted with typesetting codes and unstructured. Many data formats were incompatible and only read by their authoring applications. Shifts in system, platform and application technology could render vast document warehouses into legacy data.

The document paradigm changed with the introduction of SGML [SGML 86]. This change established a second tier of data: data describing data or metadata. The document was logically torn apart, separating content (data), structure (metadata) and formatting (metadata). This separation of document components allowed for focused document creation, as content could be created, parallel to, rather than with metadata.

There is now a wealth of information on every subject available on the Net. The possible uses of the Web seem endless, but there the technology is missing a crucial piece. Missing is a part of the Web which contains information about information - labeling, cataloging, relational and descriptive information structured in such a way that allows Web pages to be properly searched and processed in particular by computer. In other words, what is now very much needed on the Web is metadata.

This section investigates the need for data description, meta or otherwise. It explores several metadata proposals as a means of showing the impact the application of metadata has made.

3.2 Metadata Issues

The information now available on the Internet on a particular topic varies greatly in both quantity and quality. The World Wide Web has enabled users to electronically publish information accessible to millions of people relatively easily, but the ability of those people to find relevant material has decreased dramatically as the quantity of information on the Internet grows.

One emerging trend is to enable the users to describe their own material with metadata. Metadata can be used to describe an Internet resource: what it is, what it is about, where it is, how is related to what, and so on.

There are three major aspects for the deployment of metadata [Ianella 97]:

- description of resources
- production of the metadata
- use of the metadata

The first aspect addresses what set of information is to be captured by the metadata. This will depend on the type of the resource and on the purpose of the metadata. A metadata scheme must be sufficiently flexible to capture useful information about a wide variety of resources for a range of purposes.

Ideally, a single metadata scheme should be used as this minimizes the cost of using metadata. Unfortunately, it is unlikely that there will ever be agreement on a single metadata scheme and so a major aspect of metadata research is the relationship between different metadata schemes and the trade-off between the size and utility of the metadata element set.

The second aspect is the production of metadata. Metadata is essentially a summary of the data produced by various levels of "intelligence". Using humans to generate these summaries is expensive and metadata systems attempt to reduce this cost by making humans more productive by automating as much of the process as possible. The use of tools that could automatically provide this information, like DHE, would greatly advance the materialization of a metadata infrastructure.

The final aspect of metadata concerns how the metadata is accessed and used. It must be retrieved in a form that can be processed with its semantics preserved. An important use of metadata is as a mechanism for resource location in distributed networks like the Internet. Metadata can provide information for the user to identify which resources they might be interested in, with the help of relationships. Once a resource has been identified, metadata provides the information to allow the resource to be accessed.

Metadata is not new. Librarians have been cataloging books and journals for hundreds of years. The library catalogue is, in effect, metadata that is used to find books and journals about a particular subject and to retrieve them from the library shelves. Although effective, library cataloging faces a scalability problem in producing the metadata. With so many dynamic documents being published on the Web, it is not cost effective (or really possible) for librarians to professionally catalog each Web document.

3.2.1 Metadata Utility

Metadata has many uses in assisting the use of electronic and non-electronic resources on the Internet. These include:

- summarize the meaning of the data (i.e., what is the data about).
- allow users to search for the data.
- allow users to determine if the data is what they want.

- prevent some users (e.g., children) from accessing data.
- retrieve and use a copy of the data (i.e., where do I go to get the data).
- instruct how to interpret the data (e.g., format, encoding, encryption).
- help decide which instance of the data should be retrieved (if multiple formats are provided).
- give information that affects the use of data such as legal conditions on use, its size, or age).
- give the history of data such as the original source of the data and any subsequent transformations.
- give contact information about the data such as the owner.
- indicate relationships with other resources (e.g., linkages to previous and subsequent versions, derived datasets, other datasets in a sequence, and other data or programs which should be used with the data).
- control the management of the data (e.g., archival requirements, and destruction authority).

Metadata has an important role for supporting the use of electronic resources and services. However, many issues for effective support and deployment of metadata systems still need to be addressed.

Below is an example library catalogue record showing metadata about a book. It uses a structure for the elements taken from the NJIT Library Catalogue:

Material:	Book
Call Number:	QA76.76.H94 M3883 1998
Author:	McGrath, Sean.
Title:	XML by example : building E-commerce applications / Sean McGrath.
Publication:	Upper Saddle River, NJ : Prentice Hall PTR, c1998.
Description:	xlviii, 470 p. : ill. ; 24 cm. + 1 computer laser optical disc (4 3/4 in.)
Series:	The Charles F. Goldfarb series on open information management
Notes:	MP19990226
Notes:	System requirements for accompanying computer disc: Windows 95/NT; HTML 3.2 compatible Web browser; Java based tools will require a Java Virtual Machine.
Notes:	Includes index.
Notes:	RACQ19990203 CS
Subject:	XML (Document markup language)
Subject:	Electronic commerce.

Metadata is not limited to describing documents. Any resource (e.g., video, image, audio, etc) can be described with an appropriate metadata element set.

3.2.2 Metadata Model

The basic model used for metadata is known as "attribute type and value" model. Metadata is represented as a set of facts about the resource (e.g., "title", "author"). Each fact is represented as an attribute (also known as an element). An attribute contains a type (which identifies what information the attribute contains) and one or more values (the metadata itself).

For example, the attribute "<Title> XML by example : building E-commerce applications / Sean McGrath." has the attribute type "title" (indicating that this is a title) and the value "XML by example : building E-commerce applications / Sean McGrath.".

3.2.3 Metadata Standards

Metadata standards define sets of attributes that can be used to describe resources. These standards define:

- what information can be contained in the description (i.e., the set of attributes)
- which attributes are mandatory and which are optional
- what, precisely, each attribute means
- the syntax of the attribute value (i.e., rules for the format and construction of values). This might include sets of permitted values (i.e., a taxonomy).

However, there are a number of issues with the standards, not all of them are resolved. These include:

- *Accessibility of standards.* Documenting attribute sets and building the information from the standards into systems so that the computer can assist the user in constructing and using metadata.
- *Relationships between different metadata standards.* There are already many metadata standards and more will undoubtedly be created, which will lead to the situation where a resource will be described by two (or more) sets of metadata attributes. What happens if the two sets have contradictory information? Can metadata be transformed from one set to the other?
- *Extensibility of metadata standards.* It is often necessary to extend attributes sets to represent local information (e.g., linkages to existing systems). In addition, as new types of resources are defined, or new applications developed, it will become necessary to extend the metadata standards. How are these extensions published and incorporated into metadata applications?
- *Internationalization.* The range of issues involved in extending metadata from the current English model extend from presenting values so that the preferred language is presented first, to the use of attributes which have no meaning in a particular culture.
- *The linkage of data and metadata.* Metadata needs to be tightly bound to the resource it describes. The metadata must be generated at the same time (or very soon after) the resource, modified when the resource changes or is deleted.

- *Metadata is data* (at another level of abstraction). There are all the problems of storing it somewhere, finding it again, and understanding what the contents mean.

The Meta Data Coalition [MDC] and the Object Management Group [OMG] announced on April 1999 their first cooperative effort to develop metadata standards. In establishing a formal technical liaison, the MDC is now a Platform Member of the OMG, and the OMG is a member of the MDC.

The objective of this cross-membership is to provide a way for the two groups to work together on common standards, based on the belief that standards reduce confusion in the marketplace and increase efficiency for IT organizations. The first working session between the MDC and the OMG took place during the recent OMG Technical Meeting held at the end of March 1999 in Philadelphia, Pennsylvania.

Large companies currently must implement their own interfaces between software products if they want to employ a "best of breed" configuration. With the emergence of Internet-related technologies and the rapid integration of enterprise software forced by business pressures and internationalization, the cost of building and maintaining such interfaces across releases is becoming prohibitive. The definition and support of a standard for metadata exchange will go a long way in allowing companies to build business intelligence solutions and to track their metadata assets cost-effectively.

3.3 W3C: World Wide Web Consortium

The World Wide Web Consortium [W3C] was founded in October 1994 to improve the World Wide Web to its full potential by developing common protocols that promote its evolution and ensure its interoperability. It is an international industry consortium, jointly hosted by the Massachusetts Institute of Technology Laboratory for Computer Science in the United States; the Institut National de Recherche en Informatique et en Automatique in Europe; and the Keio University Shonan Fujisawa Campus in Japan. Services provided by the Consortium include: a repository of information about the World Wide Web for developers and users; reference code implementations to embody and promote standards; and various prototype and sample applications to demonstrate use of new technology.

The W3C's Technology & Society Domain Group's focus is, broadly, on establishing trust in the new medium of the Web. This is a difficult problem, involving both social and technical issues. Trust is established through a complex and ill-understood social mechanism including relationships, social norms, laws, regulations, traditions, and track records. The Technology & Society Domain Group activities are chosen to focus on specific areas that are both important and tractable. There is a core of technical issues that are required in any system that is to be trusted:

- The ability to **make statements** that have agreed upon meanings. The W3C Metadata Activity provides a means to create machine-readable statements.

- The ability to know **who made the statement** and to be assured that the statement is really theirs. The W3C Digital Signature Initiative provides a mechanism for signing metadata in order to establish who is making the machine-readable statement.
- The ability to **establish rules** that permit actions to be taken, based on the statements and a relationship to those who made the statements. The PICS Rules specification allows rules to be written down so they can be understood by machines and exchanged by users.
- The ability to **negotiate** binding terms and conditions. The now-completed JEPI project created the Protocol Extension Protocol (PEP) to provide for negotiation on the Web. Negotiation is also at the core of the Platform for Privacy Preferences Project (P3P).

3.3.1 W3C Metadata Activity

Metadata means "data about data" or "information about information" but probably more importantly now it should be taken to mean *machine understandable information, about distributed networked information*. The Metadata activity [W3C Meta] was formed in 1997 from the recognition within the Consortium of a common subtask to existing activities such as PICS and DSig at W3C, HTTP and WebDAV at the IETF, the Dublin Core and many other projects.

The Metadata activity is the architectural underpinning of many of the Technology and Society activities. W3C's work on Digital Signatures, Privacy Protection, and Intellectual Property Rights Management are all based on the

Resource Description Framework [RDF 99] work that is at the heart of the Metadata Activity. In addition, W3C is transitioning from its current metadata technology (PICS) to RDF.

3.3.2 HTML: HyperText Markup Language

The Web has evolved at a frantic rate, employing an increasing array of data formats and applications. They all require description, to ensure browser recognition. HTML [HTML 98], the fixed Web page markup language, has been unable to match the Web's growth and has proven inadequate as proprietary renderings of it have diverged. Central to this problem is the fixed and simple system of HTML data description. HTML metadata, an agent of data-description, is a crude yet functional scheme, based on the <META> element.

The lack of a standard metadata scheme has cost Web applications their intuition. Search engines often have to rely on the title of documents, stored in the <TITLE> element, for indexing Web pages, making them unable to provide extensive metadata to users, or perform accurate searches. Poor Web application functionality, caused by HTML inadequacies, has cost the Web credibility and usability. Many search engines, and other Web applications, look for recognizable metadata, in HTML Web pages, encoded in the <META> scheme. Unfortunately, search engines are more likely to find metadata referring to proprietary document generators, than to the nature of the document.

HTML metadata, which resides within the <HEAD> element, is a simple construct, relying on the NAME and CONTENT attributes of the <META>

element. Although this implementation offers some flexibility, being far from a standard scheme, its ability to describe complex document relationships is poor. The <META> scheme cannot describe complex relationships or use reuse objects and properties.

The HTML system of metadata is weak and only useful for describing high-level document properties. The current HTML metadata scheme will be abandoned in favor of a stronger, more functional scheme. A number of schemes have already been proposed, some of which have explicitly included HTML implementations

HTML Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
   "http://www.w3.org/TR/REC-html40/strict.dtd">
<HTML>
<HEAD profile="http://www.acme.com/profiles/core">
 <TITLE>How to complete Memorandum cover sheets</TITLE>
 <META name="author" content="John Doe">
 <META name="copyright" content="&copy; 1997 Acme Corp.">
 <META name="keywords" content="corporate,guidelines,cataloging">
 <META name="date" content="1994-11-06T08:49:37+00:00">
</HEAD>
 <BODY>
    <P>Hello world!
  </BODY>
</HTML>
```

3.3.3 XML: eXtensible Markup Language

The eXtensible Markup Language [XML 98] is a format designed to bring structured information to the Web, it is, in effect, a Web based language for electronic data Interchange. XML stands between HTML, designed to expressed presentation rather than semantic information, and SGML, a sophisticated tag language which separates view from content and data from metadata. XML is a subset of SGML that maintains the important architectural aspects of contextual separation while removing nonessential features.

XML is a product and catalyst of the changing document paradigm. Although a metadata scheme is out of the scope of the XML specification, the lack of one is surprising, given the XML mandate to be lean SGML for the Web. Fortunately, since XML is extensible, it allows for compliant frameworks. Many metadata initiatives have been proposed, offering XML a comprehensive metadata standard. Although they are all concerned with the same underlying constructs and problems, the depth and grasp of each proposal differs, illustrating an evolution in the conception of metadata.

Armed with a strong and standard metadata framework, XML will be a lean, yet effective, comprehensive and common document format. XML-compliant document readers need only read metadata statements to understood documents. This change may lead to users opting to set their browsers to producing summaries of document nature, choosing whether to read further, rather than the being forced to download entire documents.

Without a strong and standard metadata framework, many proprietary frameworks would be built around XML. A large percentage of those frameworks would be incompatible, making software difficult to write. This incompatibility would violate the goals of XML itself. Given the problems surrounding the metadata initiative, a standard scheme must be wrought.

XML is sometimes regarded as "semantic markup" and it is often praised for its ability to express semantic clarity through markup. What gives rise to this sentiment is a work environment within which proprietary, procedural, and implicit markup has been the norm. Someone who uses a text editor to examine an XML document - comparing it to an HTML file, to a comma-delimited text file, to Postscript, or to any document using a procedural or presentational markup language - will readily judge the XML document more meaningful with respect to the information objects represented by text. The markup itself is a form of metadata, explaining what the constituent elements are (by name), and how these information objects are structured into larger coherent units

Although there are sound technical reasons for using XML, it is clear that much of the motivation for using XML comes from the current and expected structure of the business environment related to XML. The justification for using XML contains the following technical and business factors:

- XML is already an open, platform independent, and vendor independent standard
- XML supports international character sets
- XML can represent models compliant with OMG's Meta Object Facility (MOF) framework
- XML is not linked to any one programming language or programming interface. (A range of XML application program ingterfaces (APIs) are available.)

- The cost of entry for XML information providers is low. XML documents can be created and edited by hand in a text editor. XML structure and syntax make it easy for humans to read.
- The cost of entry for automatic XML document producers and consumers is low. A growing set of tools is available for XML development, including free, commercially unrestricted high-quality tools.
- The XML approach to structured data interchange has been validated through the wide experience with XML itself and with other members of the XML family: SGML, used in high end document processing, and HTML, the predominant language of the web.
- XML is widely believed to be the next step in the evolution of the web; its uptake will enhance the ability of documents based on XML to the integrated into the information Web of the Internet
- XML is still young, but there are many well documented applications of XML, in domains such as: web commerce, publishing, repositories, modeling, databases and data warehouses, services, financial, health care, semiconductors, inventory access, and more.
- Widespread public interest in XML, leading to a substantial number of books and articles being published.

XML Example

```
<?xml version="1.0" encoding="UTF-8"?>
<nitf>
<head>
<title>Snow, Freezing Rain Batter U.S. Northeast</title>
</head>
<body>
<body.head>
<hedline>
<hl1>Snow, Freezing Rain Batter <location><country>U.S.</country>
<region>Northeast</region></location></hl1>
</hedline>
<byline>
<bytag>By Matthew Lewis</bytag>
</byline>
<dateline>
<location><city>HARTFORD</city>, <state>Conn.</state></location>
<story.date>Friday January 15 12:27 PM ET</story.date>
</dateline>
</body.head>
<body.content>

<p>Snow and freezing rain punished the <location>northeastern<country>United
States</country></location> for a second straight day on <chron
norm="19990115">Friday</chron>, causing at least five weather-related deaths,
closing airports and spreading misery from <location><city>Washington</city>,
<state>D.C.</state></location>, to
<location><country>Canada</country></location>.</p>

<p>Below a snow line that bisected
<location><state>Maryland</state></location>,
<location><state>Pennsylvania</state></location> and <location><state>New
Jersey</state></location>, a <chron norm="19990115">predawn</chron>
downpour turned road surfaces to ice. The icy buildup also brought down power
lines, leaving hundreds of thousands of people without electricity.</p>

<p><q>This is one of the most severe storms we've seen in a long time,</q> said
a <function>spokeswoman</function> for <org>Baltimore Gas and
Electric</org>. <q>We're not making any promises about when all the power will
be restored because we're still trying to find all the damage.</q></p>

<p>Some 126,000 people were left without power in
<location><state>Pennsylvania</state></location>, <location>southern
<state>New Jersey</state></location> and <location>northern
<state>Maryland</state></location>, officials said.</p>
```

```
...
<p><function>Detroit Mayor</function><person>
<name.given>Dennis</name.given><name.family>Archer</name.family></perso
n> and his staff planned to shovel snow from the porches and sidewalks of
elderly citizens on <chron norm="19990115">Friday</chron>.</p>
</body.content>
</body>
</nitf>
```

3.3.4 RDF: Resource Description Framework

The Resource Description Framework [RDF 99] provides a more general treatment of metadata. RDF is a declarative language and provides a standard way for using XML to represent metadata in the form of properties and relationships of items on the Web. Such items, known as resources, can be almost anything, provided it has a Web address. This means that users can associate metadata with a Web page, a graphic, an audio file, a movie clip, and so on.

RDF provides a framework in which independent communities can develop vocabularies that suit their specific needs and share vocabularies with other communities. In order to share vocabularies, the meaning of the terms must be spelled out in detail. The descriptions of these vocabulary sets are called RDF Schemas. A schema defines the meaning, characteristics, and relationships of a set of properties, and this may include constraints on potential values and the inheritance of properties from other schemas.

The RDF language allows each document containing metadata to clarify which vocabulary is being used by assigning each vocabulary a Web address.

The schema specification language is a declarative representation language influenced by ideas from knowledge representation (e.g., semantic nets, frames, predicate logic) as well as database schema specification languages and graph data models. One of the best-known schemas is the Dublin Core invented by the library community (their first meeting was in Dublin, Ohio, USA).

RDF uses the idea of the XML Namespaces to effectively allow RDF statements to reference a particular RDF vocabulary or "schema". Two applications might adopt the same headings and categories when it comes to organizing material. Perhaps the property address is used to mean a company location in one application, and a company's Web address in another. Potential conflicts are resolved because, through various programming mechanisms, a tag for a property name can use a short code which signals to which RDF application that tag "belongs." The XML Namespaces specification describes such mechanisms in detail and is useful not only in the context of RDF but for many other XML applications also.

There are many practical uses of RDF, for example:

- *Thesauri and library classification schemes*. These are well known examples of hierarchical systems for representing subject taxonomies in terms of the relationships between named concepts. The RDF Schema specification has exactly the features for creating RDF models that represent the logical structure of thesauri and other library classification systems.

- *Web sitemaps.* A sitemap can be seen "internally" as a description of a Web site. The RDF Schema specification provides a mechanism for defining the vocabulary needed for this kind of application. With RDF is possible to describe how one item is related to another, how one page is "a descendant" of another, and so on.
- *Description of the contents of Web pages.* This is one of the basic functions of the Dublin Core initiative. The Dublin Core is a set of 15 properties associated with bibliographic information. These can be used to describe items on the Web sufficiently well that search engines and other software can work much more efficiently. The Dublin Core Workshop series has been a major influence on the development of RDF.
- *Describing the formal structure of privacy practice descriptions.* How does a site manage personal information? Will it disclose any of this information to others? What will the user get in return? W3C's Platform for Privacy Preferences Project is working on a platform that allows users to be informed of a site's practices. Users, or software operating on their behalf, can then negotiate for a different privacy policy and come to an agreement with the site which will be the basis for any subsequent release of information. RDF may be used to describe the formal structure of privacy practice descriptions.
- *Rating systems.* These offer a way of labeling resources so that people (or computers) can filter information. RDF enables programmers to devise rating systems for any number of domains.

- *Expressing metadata about metadata.* RDF can be used to describe metadata about a given element - the date it was generated, by which organization, and so on.
- *Digital Signatures.* As the Web matures and more everyday tasks are performed online, so digital signatures will become increasingly important. RDF may be used to express information concerning what you are signing, what the significance of the signature is, the dates that the signature is valid, and so on.

RDF Example

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"/>
  <rdfs:Class rdf:ID="WebResourceMap">
    <rdfs:comment>Class for representing one or more web resources. A
WebResourceMap can contain several starting web documents and their linked
child web documents.</rdfs:comment>
    <rdfs:subClassOf resource="http://www.w3.org/TR/WD-rdf-
schema#Resource"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="WebDocument">
    <rdfs:comment>Class for representing a web document. Primary for HTML,
XML pages, but can also describe other file formats.</rdfs:comment>
    <rdfs:subClassOf resource="http://www.w3.org/TR/WD-rdf-
schema#Resource"/>
  </rdfs:Class>
  <rdf:Property ID="consistOf">
    <rdfs:comment>A WebResourceMap consists of 1 to n web
documents.</rdfs:comment>
    <rdfs:domain rdf:resource="#WebResourceMap"/>
    <rdfs:range rdf:resource="#WebDocument"/>
  </rdf:Property>
  <rdf:Property ID="linksTo">
    <rdfs:comment>A web document can link to 1 to n child web
documents.</rdfs:comment>
    <rdfs:domain rdf:resource="#WebDocument"/>
    <rdfs:range rdf:resource="#WebDocument"/>
  </rdf:Property>
```

```
 <rdf:Property ID="type">
   <rdfs:comment>Type of the WebResourceMap.</rdfs:comment>
   <rdfs:domain rdf:resource="#WebResourceMap"/>
   <rdfs:range rdf:resource="#Type"/>
 </rdf:Property>
 <!-- Possible type instances of a WebResourceMap -->
 <rdfs:Class rdf:ID="Type"/>
 <Type rdf:ID="full"/>
 <Type rdf:ID="update"/>
 <rdf:Property ID="state">
   <rdfs:comment>State of the web document.</rdfs:comment>
   <rdfs:domain rdf:resource="#WebDocument"/>
   <rdfs:range rdf:resource="#State"/>
 </rdf:Property>
 <!-- Possible state intances of a WebDocument -->
 <rdfs:Class rdf:ID="State"/>
 <State rdf:ID="unchanged"/>
 <State rdf:ID="new"/>
 <State rdf:ID="modified"/>
 <State rdf:ID="deleted"/>
</rdf:RDF>
```

3.3.5 XML Schema

While XML 1.0 supplies a mechanism, the Document Type Definition (DTD) for declaring constraints on the use of markup, automated processing of XML documents requires more rigorous and comprehensive facilities in this area. Requirements are for constraints on how the component parts of an application fit together, the document structure, attributes, data-typing, and so on. The XML Schema [XSchema 01] is addressing means for defining the structure, content and semantics of XML documents.

The purpose of a schema is to define and describe a class of XML documents by using these constructs to constrain and document the meaning, usage and relationships of their constituent parts: datatypes, elements and their content, attributes and their values, entities and their contents and notations.

Schema constructs may also provide for the specification of implicit information such as default values. Schemas document their own meaning, usage, and function.

Any application of XML can use the Schema formalism to express syntactic, structural and value constraints applicable to its document instances. The Schema formalism allows a useful level of constraint checking to be described and validated for a wide spectrum of XML applications. For applications which require other, arbitrary or complicated constraints, the application must perform its own additional validations.

The following usage scenarios describe XML applications that should benefit from XML schemas. They represent a wide range of activities and needs that are representative of the problem space to be addressed. They are intended to be used during the development of XML schemas as design cases that should be reviewed when critical decisions are made.

- *Publishing and syndication*. Distribution of information through publishing and syndication services. Involves collections of XML documents with complex relations among them. Structural schemas describe the properties of headlines, news stories, thumbnail images, cross-references, etc. Document views under control of different versions of a schema.
- *Electronic commerce transaction processing*. Libraries of schemas define business transactions within markets and between parties. A schema-aware processor is used to validate a business document, and to provide access to its information set.

- *Supervisory control and data acquisition*. The management and use of network devices involves the exchange of data and control messages. Schemas can be used by a server to ensure outgoing message validity, or by the client to allow it to determine what part of a message it understands.
- *Traditional document authoring/editing governed by schema constraints*. One important class of application uses a schema definition to guide an author in the development of documents. A simple example might be a memo, whereas a more sophisticated example is the technical service manuals for a wide-body intercontinental aircraft. The application can ensure that the author always knows whether to enter a date or a part-number, and might even ensure that the data entered is valid.
- *Use schema to help query formulation and optimization*. A query interface inspect XML schemas to guide a user in the formulation of queries. Any given database can emit a schema of itself to inform other systems what counts as legitimate and useful queries.
- *Open and uniform transfer of data between applications, including databases*. XML has become a widely used format for encoding data (including metadata and control data) for exchange between loosely coupled applications. When the exchange data model is represented by the more expressive XML Schema definitions, the task of mapping the exchange data model to and from application internal data models will be simplified.
- *Metadata Interchange*. There is growing interest in the interchange of metadata (especially for databases) and in the use of metadata registries to

facilitate interoperability of database design, DBMS, query, user interface, data warehousing, and report generation tools.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema" targetnamespace =
"http://www.example.com/baz.xsd" xmlns = "http://www.example.com/baz.xsd">
   <xs:element name="a" type="t"/>
      <xs:simpleType name="b"/>
      <xs:list base="xs:integer"/>
   </xs:simpleType>
   <xs:complexType name="t">
      <xs:attribute name="b" type="xs:string"/>
      <xs:attribute name="c" type="b" use="optional"/>
   </xs:complexType>
   <xs:complexType name="u">
      <xs:complexContent>
         <xs:extension base="t">
            <xs:choice>
               <xs:element name="d">
                  <xs:complexType>
                     <xs:sequence>
                        <xs:element name="a" type="xs:string" minOccurs="1"
                           maxOccurs="unbounded"/>
                     </xs:sequence>
                  </xs:complexType>
               </xs:element>
               <xs:element name="e" type="xs:string"/>
            </xs:choice>
         </extension>
      </complexContent>
   </xs:complexType>
</xs:schema>
```

XML document which matches the above schema

```
<baz:a xmlns:baz="http://www.example.com/baz.xsd" xs:type="baz:u" b="zero" c="1 2">
   <d>
      <a>three</a>
      <a>four</a>
   </d>
</baz:a>
```

3.3.6 Other W3C Metadata Applications

PICS: Platform for Internet Content Selection

The Platform for Internet Content Selection [PICS] consists of a suite of specifications that enable people to distribute metadata about the content of digital material in the form of "labels". These contain information about the content in simple, computer-readable form. Information can be given a label, which computers can then process in the background, filtering out undesirable material or directing users to sites that may be of special interest to them. While PICS has general applicability to labeling pages for a variety of metadata purposes, the PICS specification was originally designed to allow parents and teachers to screen out materials unsuitable for children using the Internet. Rather than simply censoring the information itself, as various legislative bodies have suggested, PICS gives responsibility to users to control personally, or to delegate control of, what they receive on their browsers

PICS Example

```
<HEAD>
 <META http-equiv="PICS-Label" content='
 (PICS-1.1 "http://www.gcf.org/v2.5"
   labels on "1994.11.05T08:15-0500"
     until "1995.12.31T23:59-0000"
     for "http://w3.org/PICS/Overview.html"
   ratings (suds 0.5 density 0 color/hue 1))'>
  <TITLE>... document title ...</TITLE>
</HEAD>
```

Dsig: Digital Signature Initiative

One element of trust is the ability to reliably associate a statement with the person or organization who made it. While the underlying cryptographic technology to accomplish this is available and widely known, it has not yet been applied to a general-purpose system for creating machine readable statements. The Digital Signature Initiative [Dsig] fills this important role by specifying how to sign statements expressed as metadata (PICS or RDF). DSig provides a plug-and-play mechanism for specifying the algorithms and data formats used for the encryption, digesting, and certificates used in the signature.

XML digital signatures are represented by the Signature element which has the following structure:

```
<Signature>
 (SignedInfo)
 (SignatureValue)
 (KeyInfo)?
 (Object)*
</Signature>
```

Dsig Example

```
<Signature xmlns="http://www.w3.org/1999/11/xmldsig-core">
 <SignedInfo ID="5">
  <CanonicalizationMethod
   Algorithm="http://www.w3.org/1999/07/WD-xml-c14n-19990729"/>
  <SignatureMethod Algorithm="&dsig;/dsa"/>
  <ObjectReference URI="http://www.mypage.com">
   <Transforms>
     <Transform Algorithm="&dsig;/null">
     <Encoding Algorithm="&dsig;/base64"/>
   </Transforms>
   <DigestMethod Algorithm="&dsig;/sha1"/>
   <DigestValue>a23bcd43</DigestValue>
  </ObjectReference>
  <ObjectReference IDREF="timestamp"
    Type="&dsig;/signatureattributes">
   <Transforms>
     <CanonicalizationMethod name="http://..."/>
   </Transforms>
   <DigestMethod Algorithm="&dsig;/sha1"/>
   <DigestValue>a53uud43</DigestValue>
```

```
    </ObjectReference>
  </SignedInfo>
  <SignatureValue>dd2323dd</SignatureValue>
  <Object ID="timestamp"
   type="&dsig;/SignatureAttributes" >
    <timestamp about="5"
     xmlns="http://www.ietf.org/rfcXXXX.txt">
      <date>19990908</date>
      <time>14:34:34:34</time>
    </timestamp>
  </Object>
  <KeyInfo>
    <keyname>Solo</keyname>
  </KeyInfo>
</Signature>
```

P3P: Platform for Privacy Preferences

This area involves the constant struggle between the need for Web content providers to gain information about their readership and the need for these individuals to control the release of this information to others. The recently initiated [P3P] Project will address the twin goals of meeting the data privacy expectations of consumers on the Web while assuring that the medium remains available and productive for electronic commerce. Following the principle of providing consumers notice of site privacy polices, and allowing users to express and act upon their privacy preferences in a flexible manner, one goal enhances the success of the other.

P3P Example

```
<POLICY xmlns="http://www.w3.org/2000/P3Pv1"
   entity="CoolCatalog, Inc.">
  <ASSURANCE-GROUP>
    <ASSURANCE org="http://www.PrivacySeal.org"
     description="PrivacySeal, a third-party seal provider"
     image="http://www.PrivacySeal.org/Logo.gif"/>
  </ASSURANCE-GROUP>
  <DISCLOSURE discuri="http://www.CoolCatalog.com/PrivacyPractice.html"
```

```
 access="none" retention="yes" change_agreement="yes"/>
 <STATEMENT>
   <IDENTIFIABLE><no/></IDENTIFIABLE>
   <CONSEQUENCE-GROUP>
     <CONSEQUENCE>a site with clothes you would
appreciate</CONSEQUENCE>
   <RECIPIENT><ours/>/RECIPIENT>
   <PURPOSE><custom/><develop/></PURPOSE>
   <DATA-GROUP>
     <DATA name="dynamic.cookies" category="state"/>
     <DATA name="dynamic.miscdata" category="pref"/>
     <DATA name="user.gender"/>
     <DATA name="user.home." optional="yes"/>
   </DATA-GROUP>
 </STATEMENT>
 <STATEMENT>
   <IDENTIFIABLE><no/></IDENTIFIABLE>
   <RECIPIENT><ours/></RECIPIENT>
   <PURPOSE><admin/><develop/></PURPOSE>
   <DATA-GROUP>
    <DATA name="dynamic.clickstream.server"/>
    <DATA name="dynamic.http.useragent"/>
   </DATA-GROUP>
 </STATEMENT>
</POLICY>
```

Electronic Commerce

An important factor in the growth of the Web is electronic commerce [W3C EC]: the ability to buy, sell, and advertise goods and services to customers and consumers. The Web is a new communications medium and, like all new media, requires us to rethink the existing solutions to age-old problems.

The main barrier to electronic commerce lies in the need for applications to meaningfully share information, not in the reliability or security of the Internet. This is due to the variety of enterprise and e-commerce systems deployed by businesses and the way these systems are variously configured and used. The problem is particularly acute when a large number of trading partners attempt to agree and define the standards for interoperation.

In order to enable consistent behaviors amongst the participants in a virtual enterprise and to allow complex interactions such as negotiation and mediation, greater levels of semantic content need to be made explicit and represented. The need for greater explicit content is exacerbated by the shift towards web automation.

Current generation web sites have been designed and developed for interaction with human users. The growth in business-to-business solutions and the integration of web sites into the mainstream IT infrastructures of business will drive the development of web sites developed specifically for interaction with other systems. Under these circumstances there will be little or no human intervention.

Whereas humans can (usually) distinguish between different interpretations of either the information content or transaction intent, these systems will only be able to operate meaningfully if the content exchanged between them, and the services they provide, carry sufficient explicit information.

XML/EDI Example

```
<!DOCTYPE Book-Order SYSTEM "edi-lite.dtd">
<Book-Order Supplier="4012345000951" Send-to="mailto:orders@sgml.u-
net.com">
 <Title>EDItEUR Lite-EDI Book Ordering</Title>
 <Order-No>967634</Order-No>
 <Message-Date>19990308</Message-Date>
 <Buyer-EAN>5412345000176</Buyer-EAN>
 <Order-Line Reference-No="0528835">
  <ISBN>0201403943</ISBN>
  <Author-Title>Bryan, Martin/SGML and HTML Explained</Author-Title>
```

```
    <Quantity>1</Quantity>
  </Order-Line>
  <Order-Line Reference-No="0528836">
    <ISBN>0856674427</ISBN>
    <Author-Title>Light, Richard/Presenting XML</Author-Title>
    <Quantity>1</Quantity>
  </Order-Line>
</Book-Order>
```

Security

An important factor in the growth of the Web is the trust that can be placed in the quality, provenance, reliability, and privacy of information available from or transferred over the Web. The Web, while relying on the underlying security offered by the Internet, has trust and security problems related to the needs of applications, and these cannot be supplied strictly at the network level [W3C Sec].

3.4 Other Organizations, Specifications, Standards and Formats

There are a number of organizations providing emerging metadata specifications and deployment infrastructures that are gaining momentum on the Internet. This section summarizes some of these metadata specifications that have a high www deployment. There are numerous other metadata sets available (e.g., IAFA Templates, GILS, etc) and deployment infrastructures (such as X.500 Directory Services). The challenge is to bring together the communities and agree to deploy flexible metadata infrastructures to support multiple (and extensible) metadata standards.

Dublin Core: Metadata for Electronic Resources

An OCLC/NCSA Metadata Workshop held in March 1995 in Dublin, Ohio [Weibel 95] identified two types of resource descriptions for networked electronic documents: automatically generated indexes used by locator services such as Lycos and WebCrawler; and cataloguing records, such as MARC, created by professional information providers. Automatically generated records often contain too little information to be useful, while manually generated records are too costly to create and maintain for the large number of electronic documents currently available on the Internet. The workshop proposed a core set of identifier elements (the Dublin Metadata Core Element Set or " Dublin Core") for "document-like objects" (DLOs), intended to mediate these extremes. "The goal of the workshop was to define a set of data elements simple enough for authors and publishers to use in describing their own documents as they put them on the Net, but useful enough to facilitate discovery and retrieval of these documents by others" [Weibel 95])

The workshop did not rigorously define what a DLO was; however "the intellectual content of a DLO is primarily text, and....the metadata required for describing DLOs will bear a strong resemblance to the metadata that describes traditional printed text" [Weibel 95]. It is clear that e.g., an electronic text, map, or image could be a DLO. The core data element set did not have to handle every type of resource that could theoretically be available on or through the network, it had only to handle document-like objects. The core element set itself is still a moving target.

The Dublin Core [DC] is a metadata element set intended to facilitate discovery of electronic resources. Originally conceived for author-generated description of Web resources, it has attracted the attention of formal resource description communities such as museums, libraries, government agencies, and commercial organizations.

The Dublin Core Workshop Series has gathered experts from the library world, the networking and digital library research communities, and a variety of content specialties in a series of invitational workshops. The building of an interdisciplinary, international consensus around a core element set is the central feature of the Dublin Core. The progress represents the emergent wisdom and collective experience of many stakeholders in the resource description arena. An open mailing list supports ongoing work.

The characteristics of the Dublin Core that distinguish it as a prominent candidate for description of electronic resources fall into several categories:

- *Simplicity*. The Dublin Core is intended to be usable by non-catalogers as well as resource description specialists. Most of the elements have a commonly understood semantics of roughly the complexity of a library catalog card.
- *Semantic Interoperability*. In the Internet, disparate description models interfere with the ability to search across discipline boundaries. Promoting a commonly understood set of descriptors that helps to unify other data content standards increases the possibility of semantic interoperability across disciplines.

- *International Consensus*. Recognition of the international scope of resource discovery on the Web is critical to the development of effective discovery infrastructure. The Dublin Core benefits from active participation and promotion in some 20 countries in North America, Europe, Australia, and Asia.
- *Extensibility*. The Dublin Core provides an economical alternative to more elaborate description models such as the full MARC cataloging of the library world. Additionally, it includes sufficient flexibility and extensibility to encode the structure and more elaborate semantics inherent in richer description standards.
- *Metadata Modularity on the Web*. The diversity of metadata needs on the Web requires an infrastructure that supports the coexistence of complementary, independently maintained metadata packages. The World Wide Web Consortium has implemented an architecture for metadata for the Web. The Resource Description Framework, or RDF, is designed to support the many different metadata needs of vendors and information providers.

The current metadata set, which was finalized in September 1998, consists of 15 elements:

- Title
- Author or Creator
- Subject and Keywords
- Description
- Publisher
- Other Contributors
- Date
- Resource Type

- Format
- Resource Identifier
- Source
- Language
- Relation
- Coverage
- Rights Management

Each element is repeatable and optional, and the entire set has been defined as *extensible*. Each Dublin Core metadata element can also have a sub-type and sub-scheme information. This provides additional semantics to the values of the metadata.

Dublin Core Example

Element	Content	Scheme
DC.Title	Building Web Applications with UML	
DC.Creator.CorporateName	Jim Conallen	
DC.Creator.Jurisdiction	United States	BEP App. 4.1
DC.Subject.Industry	ALL	BEP App. 4.2
DC.Subject.Topic	Web site development	BEP App. 4.3
DC.Description	Guide to building robust, scalable, and feature-rich web applications using proven object-oriented techniques	
DC.Publisher	Addison-Wesley	
DC.Date	01-DEC-1999	BEP App. 4.5
DC.Type	Didactic	BEP App. 4.6
DC.Format	text	BEP App. 4.7
DC.Identifier	http://www.awl.com/cseng/	BEP App. 4.8
DC.Language	en	RFC1766
DC.Coverage.PlaceName. Operation	United States	BEP App. 4.1
DC.Rights.Text	© Addison Wesley Longman, Inc.	

3.4.1. Warwick Framework

The Dublin Core working group also recognized that no single set of elements will satisfy all metadata requirements on the Internet. An infrastructure was needed to support any metadata element set. This infrastructure is called the Warwick Framework and is a container architecture for aggregating logically, and perhaps physically, distinct packages of metadata [Lagoze 96]. The architecture allows separate administration and access to metadata packages and proposes implementations of the Framework in HTML, MIME, SGML, and distributed objects.

Warwick Framework Example

```
<!DOCTYPE container system "warwick.dtd">
<container name="example">
 <indirect uri="http://foo.bar.com/huh">
 <package name="admin">
  <metadata Name="date-created">12/31/95</metadata>
  <metadata Name="last-revised">4/5/96</metadata>
 </package>
 <DublinCore>
  For this example, just some text. A DTD for the Dublin
  core is being developed, and the content here should conform
  to it in real use.
 </DublinCore>
 <package name="misc" notation="RFC-822">
  From: daniel@acl.lanl.gov (Ron Daniel)
  Subject: Metadata tagging schemes
 </package>
</container>
```

3.4.2. MDC: MetaData Coalition

The MetaData Coalition [MDC] was founded in 1995 to develop and provide standardized metadata exchange; the coalition introduced the MetaData Interchange Specification [MDIS 97] in 1996. Recently the MDC completed the technical review of the MDC-OIM [OIM 98], a technology-independent and vendor-neutral information model describing the structure and semantics of metadata.

3.4.2.1. MDIS: MetaData Interchange Specification. The heart of the MetaData Interchange Specification is the core set of components that represents the minimum common denominator of metadata elements and the minimum points of integration that must be incorporated into tool products for compliance [MDIS 97]. Compliance with the MDIS requires support for all relevant core set components and integration points in accordance with the approved specifications.

The MDIS also provides for an approved set of optional/extension components that are relevant only to a particular type or class of tool or a specific application or architecture. Because these are used by more than one tool or application, they can and should conform to the specification definition and set of access parameters, but because they are not generic across all tools, architectures, or applications they would not be eligible for the core set, nor required for compliance.

The members of the MDC agreed upon initial MDIS goals, including:

- Creating a vendor-independent, industry-defined and -maintained standard access mechanism and standard application programming interface (API) for metadata;
- Enabling users to control and manage the access and manipulation of metadata in their unique environments through the use of interchange specification-compliant tools;

- Allowing users to build tool configurations that meet their needs and to incrementally adjust those configurations as necessary to add or subtract tools without impact on the interchange specification environment;
- Enabling individual tools to satisfy their specific metadata access requirements freely and easily within the context of an interchange model;
- Defining a clean, simple interchange implementation infrastructure that will facilitate compliance and speed adoption by minimizing the amount of modification required to existing tools to achieve and maintain MDIS compliance; and
- Creating a process and procedure not only for establishing and maintaining the MDIS but for extending and updating it over time as required by evolving industry and user needs.

MDIS Example

```
BEGIN HEADER
 CharacterSet "FRENCH"
 ExportingTool "IEF Composer"
 ToolVersion "3.1"
 ToolInstanceID "5"
 MDISVersion "1.1"
 Date "1996-03-15"
 Time "14.32.18"
END HEADER
BEGIN ApplicationData
 Tool "tool 1"
 BEGIN ToolAppData
  up to each tool
 END ToolAppData
 Tool "tool 2"
 BEGIN ToolAppData
  kw val
  kw val
 END ToolAppData
END ApplicationData
```

```
BEGIN DATABASE
 Identifier "001"
 ServerName "NEWTON"
 DatabaseExtendedType "AIX1.0"
 OwnerName "HRADMIN"
 DatabaseName "PAYROLL"
 DateCreated "1992-12-02"
 TimeCreated "23.12.15"
 DateUpdated "1996-03-10"
 TimeUpdated "08.00.00"
 BriefDescription "DB2/MVS payroll database at Newton site"
 BEGIN ApplicationData
  Tool "DXT"
  BEGIN ToolAppData
   CREATE DXT FILENAME=PAYROLL, DESC="DB2/MVS payroll
    database at Newton site"ACCESS =GDI,GDIEXIT=GDIDB2S,
   GDIXTYPE=SELECT
  END ToolAppData
 END ApplicationData
 DatabaseStatus "PRODUCTION"
 DatabaseType "RELATIONAL"
 BEGIN RECORD. . . .
END DATABASE
```

3.4.2.2. OIM: Open Information Model. The Open Information Model [OIM 98] is a set of metadata specifications to facilitate sharing and reuse in the application development and data warehousing domains. OIM 1.0 consists of over 200 types and 100 relationships, described in UML (Unified Modeling Language) and organized in easy-to-use and easy-to-extend subject areas. The metadata model developed by over 20 industry-leading companies, is based on industry standards such as XML, SQL, COM, Java, and UML, and has been reviewed by over 300 companies.

The overall purpose of the Open Information Model is to support tool interoperability via a shared information model, across technologies and across companies. The OIM is designed to encompass all phases of a project's development life cycle, from analysis through deployment. Computing technologies as diverse as CASE, component, application, intranet, database, and data warehousing will be supported.

The Open Information Model is grouped into subject areas addressing domains of varying complexity and generality. For example, the subject areas range from the database model, which describes database schemas in general, to the SQL Server and Oracle subject areas, which describe the unique features of those individual implementations. OIM 1.0 targets three main areas:

- Object-oriented Analysis and Design
- Component Description and Specification
- Database Schema and Data Warehousing

OIM Example

```
<?xml versio ="1.0" ?>
 <oim:Transfer xmlns:oim= "http://www.mdcinfo.com/oim/oim.dtd"
        xmlns:dbm= "http://www.mdcinfo.com/oim/dbm.dtd" >
  <dbm:Catalog id= "_1"  name= "sales"  comments= "Sample catalog" >
   <dbm:CatalogSchemas>
    <dbm:Schema id= "_2"  name= "dbo" >
    <dbm:SchemaTables>
     <dbm:Table id= "_3"   name= "Customer" >
      <dbm:Column SetColumns>
       <dbm:Column  id= "_6"   name= "CustomerID"  IsNullable= "0"  />
       <dbm:Column  id= "_7"   name= "Nname"  IsNullable= "0"  />
       <dbm:Column  id= "_8"   name= "Address"  IsNullable= "1"  />
       <dbm:Column  id= "_9 "  name= "Phone"  IsNullable= "1"  />
      </dbm:Column SetColumn s>
     </dbm:Table>
     <dbm:Table id= "_4"   name= "Order"   stimatedRows= "10000" >
      <dbm:Column SetColumns>
       <dbm:Column  id= "_10"   name= "CustomerID"  IsNullable= "0"  />
       <dbm:Column  id= "_11"   name= "OrderID"  IsNullable= "0"  />
       <dbm:Column  id= "_12"   name= "Date"  IsNullable= "1"  />
      </dbm:Column SetColumns>
     </dbm:Table>
    <dbm:Table id= "_5"  name= "OrderItem"  stimatedRows= "100000" >
     <dbm:Column SetColumns>
      <dbm:Column  id= "_13"  name= "CustomerID"  IsNullable= "0"  />
      <dbm:Column  id= "_14"  name= "OrderID"  IsNullable= "0"  />
      <dbm:Column  id= "_15"  name= "LineNo"  IsNullable= "0"  />
      <dbm:Column  id= "_16"  name= "Description"  IsNullable= "1"  />
      <dbm:Column  id= "_17"  name= "Quantity"  IsNullable= "0"  />
      <dbm:Column  id= "_18"  name= "UnitPrice"  IsNullable= "0"  />
     </dbm:Column SetColumns>
     <dbm:TableU iqueKeys>
      <dbm:U iqueKey id= "_19"  name= "PK_OrderItem"  IsPrimary= "1" >
       <dbm:KeyColumns>
        <dbm:Column  href= "#_14"  />
        <dbm:Column  href= "#_15"  />
       </dbm:KeyColumns>
      </dbm:UniqueKey>
     </dbm:TableUniqueKeys>
    </dbm:Table>
   </dbm:SchemaTables>
  </dbm:Schema>
  </dbm:CatalogSchemas>
 </dbm:Catalog>
</oim:Transfer>
```

3.4.2.3. XIF: XML Interchange Format. The Metadata Coalition, in cooperation with the leading independent repository vendors, recently announced the availability of XML as interchange format for metadata described by the Open Information Model [OIM 98].

XML is the native interchange format for metadata described by the OIM. OIM is a set of metadata specifications to facilitate sharing and reuse in the application development and data warehousing domains. OIM 1.0 consists of over 200 types organized in easy-to-use and easy-to-extend subject areas.

The representation of OIM instances based on XML provides a powerful and easy-to-implement mechanism to exchange metadata between multiple heterogeneous repositories offered by different vendors. This is the first time that enterprise customers are able to interchange and integrate application development and data warehousing metadata using a standard encoding format and a broadly accepted industry standard information model.

The XML Interchange Format [XIF 99] for OIM enables the exchange of metadata between heterogeneous repositories and tool stores. As an open, industry-standard model accommodating metadata of software development and data warehousing tools, the Open Information Model provides a content-rich, yet vendor-neutral specification of metadata. Vendors supporting XIF will be able to import and export metadata, such as analysis and design models, component descriptions, and data warehousing transformations.

Besides supporting the interchange of metadata across multiple repository tools, XIF provides third-party vendors with an easy way to populate repository databases with data. For example, a third-party tool can insert OIM-compliant instances into Microsoft Repository by creating an XML file, then using the Microsoft XIF Import/Export Utility.

XIF Example

```
<dbm:Table xif:id="_1">
   <xif:name>Addresses</xif:name>
   <uml:members xif:size="2">
      <dbm:Column xif:id="_2" ordinal="1">
         <xif:name>Zip Code</xif:name>
         <dbm:type>Numeric</dbm:type>
      </dbm:Column>
      <dbm:Column xif:href="#_3" xif:ordinal="2">
      </dbm:Column>
   </uml:members>
</dbm:Table>
```

3.4.3. OMG: Object Management Group

The Object Management Group [OMG] has provided leadership in metadata management starting with issuance of the Repository RFI in 1995, which led to the OMG distributed repository architecture definition in 1996. The Meta Object Facility [MOF 97] was adopted by the OMG in 1995 and has been refined through the OMG's open, vendor-neutral standards process. The Unified Modeling Language [UML 97] was adopted in 1997. More recently, the OMG embraced W3C XML with the adoption of the XML Metadata Interchange [XMI 98]. These three standards, UML, MOF and XMI, form the foundation of the OMG's modeling and metadata management architecture. This architecture is designed to be technology- and middleware-neutral to foster rapid consensus in the industry in metadata standardization.

3.4.3.1. XMI: XML Metadata Interchange. XMI (XML Metadata Interchange) [XMI 98] has been created within the technology development process of the Object Management Group, with the aim of allowing developers of distributed systems to share object models and other structured data, created using OMG technology, over the Internet.

The main purpose of XMI is to support the exchange of metadata between modeling tools based on the OMG UML (an object based modeling language), and repositories based on the OMG MOF (a standard for describing metadata structures, such as data describing UML models, and repository interfaces). XMI is an integral part of the OMG architecture, which is aimed at supporting interoperation of information systems within diverse and distributed installations.

XMI uses XML so that XMI data exchanges can take advantage of the extensive technology development surrounding XML and the Web, rather than creating an alternative competing standard for structured data exchange.

XMI is made up of:

- Rules for generating XML Document Type Definitions (DTDs) from MOF based metamodels (these metamodels are descriptions of classes of metadata; these classes of metadata may include the specifications of the metamodels themselves, to as many levels as required)
- Rules for generating XML Documents from MOF based metadata, and MOF based metadata from XML Documents
- Design guidelines for XMI-related DTDs and XML data
- Actual DTDs supporting UML and MOF

XMI is likely to be useful where there is additional information (for example, constraints on data values, or constraints on data structure) which is beyond the expressive capability of XML, and which is essential for a full exchange of meaningful information. XMI was designed to be used for exchanging information not only between repositories driven by OMG specifications. XMI supports the exchange of any kind of structured data that can be described using MOF.

So what is the added benefit of XMI over plain XML? - XML itself provides for structured data, and syntactic specifications of the structure of that data; XMI provides a mechanism for encoding many levels of information modeling into the single syntactic interchange structure of XML.

The core of XMI is a pair of parallel mappings: between MOF metamodels (model descriptions) and XML DTDs; and between MOF metadata (model instances) and XML documents. Any repository or tool that can encode and decode XMI streams can exchange structured information with other repositories or tools with XMI capability.

XMI Example

```
<Model xmi.id="i00000001">
 <name>model1</name>
 <ownedElement>
  <Class xmi.id="i00000002">
   <name>class1</name>
    <feature>
     <Attribute xmi.id="i00000003">
      <name>attribute1</name>
      <type><integer/></type>
     </Attribute>
    </feature>
  </Class>
 </ownedElement>
</Model>
```

3.5. Metadata Role in Indexing Services

One of the major reasons for moving towards author-described resources with metadata is to try and provide more effective indexing services for the public. Current Web indexing services will attempt to summarize a document by analyzing the HTML code and producing a summary. This does not always produced effective results, and coupled with the enormous mass of Web documents, the end result usually ends up disappointing the information seeker.

Some of the current Web indexing services (in particular, AltaVista and Infoseek) are now currently supporting a limited metadata set (two elements; *Description* and *Keywords*). When these services index a www site, it will look for these META tags, and index the document based on the author-supplied list of *Keywords* (it will still try to index the rest of the document, but the keyword list takes precedence). The *Description* field is used in the display of the returned result-set.

When metadata becomes more common (either embedded in documents, such as the META tag in HTML files, or from a separate metadata repository) and indexing services start to concentrate on indexing this information, we should see a marked increase in the effectiveness of information retrieval. The author-generated metadata (or even semi-automated) will add a higher level of quality.

3.6 Meta-Information

The ability of the Internet and the World Wide Web protocols to integrate services from different organizations is transforming business practice more rapidly than ever before. Many businesses have recognized that connectivity to the Internet, and effective exploitation of available Internet services and data, has become a major factor in their overall competitiveness. Relationships with suppliers, partners, and customers will soon be mediated almost exclusively using Internet technology, and integration is becoming deeper, broader and more seamless than was ever deemed possible.

Over the next few years, the role of the IT/IS department within many businesses will change out of all recognition. The feasibility to deliver a greater diversity of networked services over public networks is creating the opportunities for new models of how information systems are conceived, implemented and sourced. The time is fast approaching when it will be possible to create, upon demand, the necessary information infrastructure to support complex virtual enterprises, and to dismantle these when they are no longer needed.

Already there is evidence of the emergence of a new kind of Internet commerce providers, able to source IT/IS services to others, over the Internet. Hot technology start-ups are examining online delivery models of various types and are looking to offer a wide range of digital online services to businesses hungry to reduce overheads and gain competitive advantage. Adoption of such services will depend upon stability in service offerings and meaningful commercial arrangements under which services will be offered. If this can be achieved, expect many businesses to concentrate upon their core competence and procure IT/IS services as required via the net, integrating them into their wholly owned infrastructures. Business will integrate 3rd party services with their owned infrastructure to an extent commensurate with the degree to which these new services become, or are perceived to be, business critical.

Forward thinking organizations are beginning to organize, standardize and stabilize offered services in order to create and maintain sustainable computer-mediated relationships with e-business partners. Part of the current excitement surrounding metadata expressed with XML is that the SGML, EDI and Internet communities expect it to be able to bring a sense of order to electronic trading relationships. After all, in many cases, today's best practice in electronic commerce is little more than the integration of legacy system functionality into the Web architecture. XML (metadata) is changing the landscape forever and giving the digital entrepreneurs a myriad of new ways to carve a niche for themselves in the online landscape.

Once the potential for site to site interactions takes hold, combined with a rich set of brokering, mediation, negotiation, co-ordination and notification services, the complexity of the digital space will multiply exponentially. Process and information modeling technologies will become an essential part of the toolkit for enterprises wishing to take up a role in the medium. XML will be both a solution and purveyor of additional complexity, helping the development and implementation of the Semantic Web.

[Grønbæk 00] introduces an approach to utilize open hypermedia structures such as links, annotations, collections and guided tours as metadata for Web resources. The paper introduces an XML based data format, called Open Hypermedia Interchange Format - OHIF, for such hypermedia structures. OHIF resembles XLink with respect to its representation of out-of-line links, but it goes beyond XLink with a more rich set of structuring mechanisms, including e.g., composites. Moreover OHIF includes an addressing mechanisms (LocSpecs) that goes beyond XPointer and URL in its ability to locate non-XML data segments. By means of the Webvise system, OHIF structures can be authored, imposed on Web pages, and finally linked on the Web as any ordinary Web resource.

Following a link to an OHIF file automatically invokes a Webvise download of the metadata structures and the annotated Web content will be displayed in the browser. Moreover, the Webvise system provides support for users to create, manipulate, and share the OHIF structures together with custom made web pages and MS Office 2000 documents on WebDAV servers. These Webvise facilities goes beyond ealier open hypermedia systems in that it now allows fully distributed open hypermedia linking between Web pages and WebDAV aware desktop applications. [Grønbæk 00] describes the OHIF format and demonstrates how the Webvise system handles OHIF. Finally, it argues for better support for handling user controlled metadata, e.g., support for linking in non-XML data, integration of external linking in the Web infrastructure, and collaboration support for external structures and metadata.

3.7 Summary

Metadata is data that describes data. There is a need to expedite the access and use of this data. Currently, computer science lacks a standard metadata framework, broad enough to describe any kind of resource.

Resources are required to evolve the Web from their current state to one where they are organized, catalogued and effectively searchable. The Semantic Web Activity has been recently established to lead both the design of enabling specifications and the open, collaborative development of technologies that support the automation, integration and reuse of data across various applications.

The document paradigm changed with the introduction of. This change established a second tier of data: data describing data or metadata. The document was logically torn apart, separating content (data), structure (metadata) and formatting (metadata).

There are three major aspects for the deployment of metadata: description of resources, production, and use.

Ideally, a single metadata scheme should be used as this minimizes the cost of using metadata. Unfortunately, it is unlikely that there will ever be agreement on a single metadata scheme.

Using humans to generate summaries is expensive, and metadata systems attempt to reduce this cost by making humans more productive by automating as much of the process as possible. The use of tools that could automatically provide this information, like DHE, would greatly advance the materialization of a metadata infrastructure.

Metadata has many uses in assisting the use of electronic and non-electronic resources on the Internet. These include: summarizations, search, filtering, retrieval, interpretation, versioning, history, contact, control, and relationships.

The basic model used for metadata is known as "attribute type and value" model. Metadata standards define sets of attributes that can be used to describe resources.

For the W3C Metadata Activity, metadata should be taken to mean machine understandable information, about distributed networked information.

The Metadata activity is the architectural underpinning of many of the W3C's activities

XML is a format designed to bring structured information to the Web. XML-compliant document readers need only read to metadata statements to understand documents. XML is regarded as "semantic markup.

RDF provides a more general treatment of metadata. RDF is a declarative language and provides a standard way for using XML to represent metadata in the form of properties and relationships of items on the Web.

The XML Schema addresses means for defining the structure, content and semantics of XML documents. The purpose of a schema is to define and describe a class of XML documents by using constructs to constrain and document the meaning, usage and relationships of their constituent parts.

The main barrier to electronic commerce lies in the need for applications to meaningfully share information. In order to enable consistent behaviors amongst the participants in a virtual enterprise and to allow complex interactions such as negotiation and mediation, greater levels of semantic content need to be made explicit and represented. The need for greater explicit content is exacerbated by the shift towards web automation.

There are a number of organizations providing emerging metadata specifications, such as: Dublin Core, Warwick Framework, MetaData Coalition, MetaData Interchange Specification, Open Information Model, and XML Interchange Format.

UML, MOF and XMI, form the foundation of the Object Management Group 's modeling and metadata management architecture. This architecture is designed to be technology- and middleware-neutral to foster rapid consensus in the industry in metadata standardization.

The ability to find relevant material has decreased dramatically as the quantity of information on the Internet grows. One of the major reasons for moving towards author-described resources with metadata is to try and provide more effective indexing services for the public.

When metadata becomes more common, and indexing services start to concentrate on indexing this information, there will be a marked increase in the effectiveness of information retrieval

DHE makes extensive use of metadata. The automatic link generation aspect of the DHE relies on the inclusion of semantic metadata for each pre-defined element present in a virtual document. The term Meta-Information is used in this research to make clear when the metadata includes relationships.

CHAPTER 4

DYNAMIC HYPERMEDIA ENGINE OVERVIEW

DHE automatically generates links at run-time for each of those elements with relationships and metadata. Such elements need to be previously identified using a methodology like the Relation-Navigation Analysis (RNA) [Yoo 00]. DHE also constructs more sophisticated navigation techniques not often found on the Web (e.g., guided tours, overviews, structural query) on top of these links. The metadata, links and navigation, as well as annotation features, supplement the application's primary functionality.

The Dynamic Hypermedia Engine (DHE) Project aims to develop a hypermedia engine, which runs in parallel with other applications and dynamically gives those applications hypermedia functionality. Dynamically means that there is a run-time mapping of the application information and relationships to hypertext objects, within both the system and external to it. An important feature to remark is that this mapping keeps changes to the application itself to a minimum. The mapping is performed with the help of logical rules called Bridge Laws.

Many applications being brought to the Web have a well-defined architecture [Bieber 95b]. A primary portion includes a computational section and a secondary portion includes the wrapper section that grabs the output of the computational part and formats it into HTML thus allowing it to be displayed on the browser, which is the final interface to the application. Based on this architecture, DHE intercepts the communication between the application's computational section and its interface section, automatically detecting the location of interrelationships in messages and documents based on the knowledge of the application's internal structure. Bridge Laws provide the mapping between the structural relationships of the application and the hypertext features. Therefore, not only does every anchor essentially lead to a set of all relationships inside the application, but also leads to any user-specified annotation and any other relationships external to the application.

This research builds on prior success. Dr. Michael Bieber's team have incorporated hypertext functionality using Bridge Laws to a model management analysis tool called TEFA as proof-of-concept prototype [Bieber 97c]. However, this uses a standalone, non Web-based hypermedia system on the PC called Microcosm [Davis 92] as its basic platform. It is possible can extend Microcosm by adding new modules (called "filters") to the existing module base. That prototype implemented the computational hypertext engine as a set of filters for TEFA written in Prolog and C.

4.1 Features

One of DHE's main objectives is to provide a testing environment for hypermedia functionalities. This environment should have the following features:

- Provide a [Web] interface
- Automated metainformation and hypermedia functionality
- Support for Virtual Documents
- Mapping rules to represent an application internal structure
- Modular architecture

- Wrappers to integrate possibly distributed applications with minimal changes
- XML as basic format for coding metaInformation

4.2 Architecture

DHE's goal is to provide full hypermedia functionality to computational applications or 'Dynamically-Mapped Information Systems' (DMIS) with minimal or no changes to them. DHE will serve any DMIS and user interface (UI) that has implemented an appropriate wrapper. To integrate a new DMIS or UI, it is necessary to develop a wrapper and a knowledge base (i.e., identify elements, bridge laws, metadata, etc.) for it. Thus, to provide a DMIS application with hypermedia support, the developer only has to declare the contents of its DMIS wrapper and knowledge base. This may prove straightforward or complex. But in any case it only must be done one time to apply to any DMIS instance [Whitehead 97].

To fully support the broad range of information systems found in organizations, the hypermedia engine must support multiple applications and users. Some researchers envision that hypermedia could even integrate independent applications for interrelated tasks and processes.

The DHE 1.0 architecture consist of several well-defined and separate processes, each possibly running on a different platform:

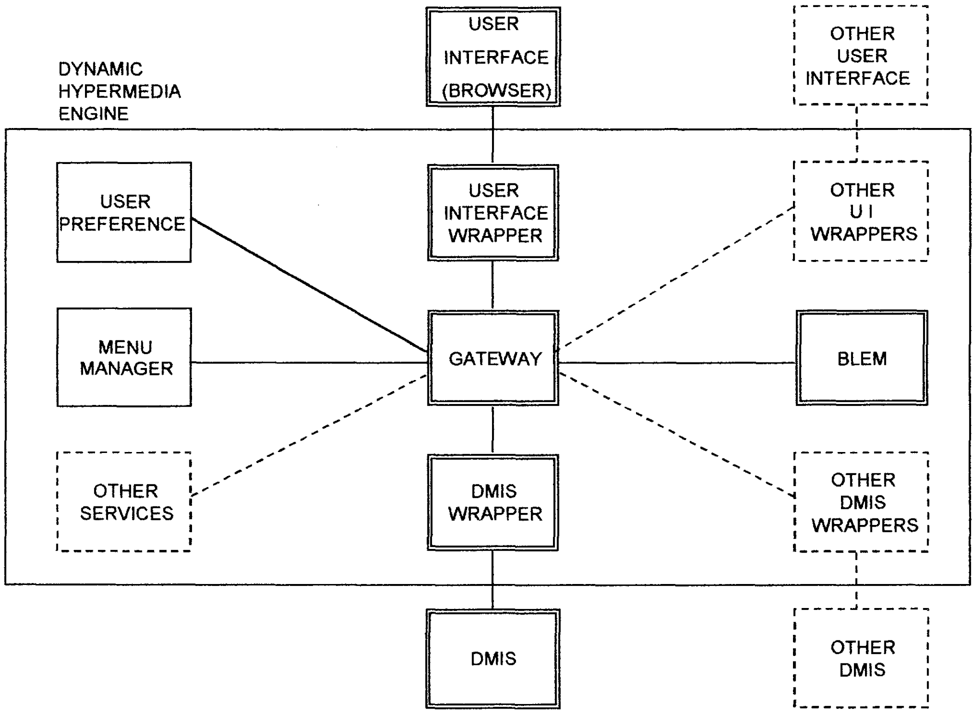


Figure 4.1 DHE 1.0 Architecture

- *User Interface* (UI), which usually runs on the user's computer (web browser).
- *User Interface Wrapper* (UIW), serves three important functions: First, it translates the displayable portion of the internal messages, from the DHE's standard format to a format the UI can process, and vice versa. Second, it handles the communication between the UIW and the UI. Third, it implements any functionality the Engine requires on the UI, which the UI cannot provide itself.
- *Gateway* (GW), enables the communication between the Engine modules and works as the router for all the DHE internal messages. All Engine

messages pass through the GW, which then redirects them to the appropriate module.

- *Traversal Path Manager* (TPM), stores the path each message has to follow. Every time a message reachs the Gateway, the TPM determines its next destination.
- *Log Manager* (LM), keeps a log of all messages being transmitted inside of the engine.

- *Bridge-Law Element Mapper* (BLEM), maps the application data and relationships to hypertext objects at run-time. BLEM maps the element instances in the virtual document to the global element types (classes), and finds the links for them. Once the links are produced they are sent through to the user interface wrapper.
    - *Link Bridge Laws* - These are used to identify links. A Link Bridge Law maps the element instances present in the DMIS output to the global DMIS elements stored in BLEM and produces the links for them. Once the links are produced they are sent to the user through to the User Interface Wrapper.
- *Dynamically Mapped Information System Wrapper* (DMISW), as the UIW, it manages the communication between the DMISW and the application system, translates the user requests from the DHE internal format to the application API, receives the output from the DMIS and converts it to the Engine format. The DMISW also identifies and marks the elements within the DMIS output to which hypermedia components are mapped.

- *Dynamically Mapped Information System* (DMIS), an application system that dynamically generates the data requested by the user.

4.2.1 Gateway

The main objective of Gateway module of DHE is to enable and manage communication interfaces between different engine modules. This module is basically responsible for passing engine messages reliably and efficiently to the destination modules. The Gateway module will receive a message, determine where it needs to route it next, and log the message into a database.

Functionality Overview

- Allow different engine modules to register.
- Keep a registry of active modules.
- Receive messages from different engine modules.
- Send messages to other modules.
- Send and receive messages simultaneously.
- Look into Gateway data base: Traversal Path Manager (TPM) table and find the next destination module.
- Provide an abstract mechanism to other engine modules for the creation modification, access and manipulation of messages.
- Provide management and debugging interface. This interface should perform: display all currently active modules and trace the processing of a message inside the Gateway.

- Log every message between two modules into a database passing through the Gateway.
- Create User Interfaces to maintain Traversal Path Manger Table (TPM) and log database.
- Constraints

4.2.1.1 Traversal Path Manager (TPM). The current DHE architecture needs to support asynchronous communication between the modules, a mechanism that the Gateway uses to determine if the message must go to other intermediate points before the message's final destination. The Traversal Path Manager's parameters are defined in a database that informs the Gateway which module is the next recipient of the message. The individual DHE module needs to be concerned with the message's final destination only, and not with the intermediate points. The TPM includes a user interface program to maintain its tables. It also contains an utility class used by Gateway to query the TPM table to determine the next destination module.

The XML message passed between modules will always contain the following elements: Origin, Current, Destination, UserID, Key and MsgBody. For example,

```
<MsgType>Display Contents</MsgType>
<Origin>DMIS</Origin>
<Current>BLEM</Current>
<Destination>UIW</Destination>
<UserID>1234567890</UserID>
<Key>testing1234567890</Key>
<MsgBody> .... </MsgBody>
```

The elements Origin, Current and Destination will have a DHE module id as value. The Gateway module will decide what is the next destination of the message based on the value of the Origin, Current, Destination, and MsgType fields.

4.2.1.2 Log Manager (LM). All DHE modules communicate with each other via the Gateway. A module sends a document to the Gateway, and the Gateway usesing the Traversal Path Manager to send the message to the next module. This module allows the Gateway to log the message into a database table. The Gateway uses the Log Manager facility to log the message as soon as it receives it. Once it determines the message's next destination, it updates the database table with the corresponding module id. The LM also includes a user interface to query and view the messages logged by the Gateway.

4.2.1.3 Engine Message. This class acts as a facilitator to the DHE project and it's modules by providing a mechanism to build XML messages. XML messages are built using the Document Object Model [W3C DOM] level 1. The Engine Message class provides methods to build, parse and debug XML tree-based documents, thus individual modules do not need to deal with implementation details of DOM. The Engine Message class is independent of Gateway; it provides a mechanism to extract XML document from an Engine Message, by which all modules communicate.

4.2.2 Bridge Laws Element Mapper (BLEM)

There are three kinds of basic hypertext objects in DHE: nodes, links and anchors. Nodes are elements of interest. They include elements inside DMISW and documents passed. Links represent relationships among two or more nodes. Thus nodes are the endpoints of links.

In the Hypermedia Engine context, a 'document' is the dynamically generated output of an Engine Module or DMIS that should be displayed by a User Interface. The kind of elements that can be considered as 'objects' or element of interest in a document are: a character or set of adjacent characters, a word or set of adjacent words, a phrase or set of adjacent phrases, a paragraph or set of adjacent paragraphs, a table, a figure, an image, etc., as well as any adjacent combination of these, including the document as a whole.

It is possible to have links or comments associated with a node. Nodes have meta-information available about them that the UIW displays in the meta-information frame. Any object inside the system could be treated as a node; in other words any object can be an "element of interest" at one point. This means that any object can have meta-information displayed about it. Any object can have a link to it. So DHE needs to display lists of links and meta-information about objects or elements of interest in the browser frames. When the user asks for any more information about element, BLEM provides the list of links and meta-information available in BLEM.

Links in the BLEM are defined in terms or 'Bridge Laws' [Bieber 92]. A bridge law "bridges" two different domains, i.e.,; it takes an object and maps it from one domain to another. In the engine project, one domain is the DMIS's environment and the other is the hypertext environment. Link bridge laws map between relationships and hypertext links. Each link bridge law represents one relationship for one DMIS element. Therefore, BLEM generates a hypertext anchor for each element that has a relationship available, and a hypertext link for each of these relationships.

Bridge laws also provide a mechanism for modules to add a command to the list of links. When the user selects an anchor and requests a list of links, bridge laws could also declare commands to appear in that list of links.

Functionality Overview

- *Receive an XML message from another Module*, through the Gateway.
    - Using the method(s) developed with the Gateway module to receive messages.
- *Extract the content of an XML message*
    - Use the methods developed with the Gateway module to read the content of message to parse it for message processing.
    - The content could be an HTML/Text document, menu, list of links, frame command, etc.
    - The content might include 'marked' element.
    - There may be Meta-information for each 'marked' element and for the content as a whole.
    - The message should contain Request(s)/Action(s) to be executed by the BLEM.
    - Search for meta-information for a particular element instance and/or element type.

  - Search for Bridge Laws for a particular element instance and/or element type.
  - Mark the elements for which there are associated Bridge Laws.
  - Include the meta-information found for each element in the message.
  - Return a list of links for a particular element and/or element type.
- *Send an XML message*
  - The message must travel through the Gateway to its final destination.
  - Using the method(s) provided by the Gateway module to send messages.
  - The Traversal Path Manager in the Gateway should indicate if the message must go to other intermediate points before the final destination.
- *Provide an Interactive User Interface*. This interface should perform the following tasks:
  - Display information of currently stored Bridge Laws.
  - Display information of currently stored Meta-information.
  - Manually Add/Update/Delete Bridge Laws to the database.
  - Manually Add/Update/Delete Metadata to the Database.

4.2.3 User Interface Wrapper

The User Interface Wrapper (UIW) is one of the core components of the Hypermedia Engine. Its main function is to enable the communication between the user interface and the DHE. The UIW is responsible for handling and processing requests from different users, and sending them to the proper modules for processing and retrieving information. In this particular instance of UIW, the user interface is defined as a web browser and the output format as HTML. All the information currently sent to the browser is coded in HTML format. HTML was selected to make DHE accessible from any web browser, since it is the web standard more closely followed by the browsers development companies. On the other hand, all communications among DHE modules are coded using the eXtensible Markup Language (XML). The UIW marks the documents embedded inside the XML messages using browser understandable HTML. It also does the reverse, by transforming HTTP requests into XML messages after a user clicks on a hyperlink or a form submission button on the user interface.

The UIW was implemented using Java Servlets. In general, servlets are modules that extend request/response-oriented servers, such as Java-enabled web servers. Servlets can be used to send and receive Hypertext Transfer Protocol (HTTP) requests and responses. Servlets, furthermore, can handle multiple users simultaneously and efficiently with the ability to keep track of each user's session. All these benefits were needed in order for the UIW to perform its tasks.

In Sun's own description for servlets, servlets are to servers what applets are to browsers. Unlike applets, however, servlets have no graphical user interface.

The UIW could have also been implemented using other technologies, like the Common Gateway Interface (CGI), but servlets are much more efficient than

CGI. Servlets 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. Servlets also address the problem of doing server-side programming with platform-specific APIs: they are developed with the Java Servlet API, a standard Java extension. Furthermore, CGI generates a new process for each request and puts lots of overhead and processing load on servers. On the other hand, servlets are thread based and does not generate any extra overhead and puts less workload on servers.

Functionality Overview

This section describes the overall functionality of the User Interface Wrapper:

- *Receive Information from the User Interface (Web Browser):* The UIW is responsible of receiving the information entered by the users. This information would be submitted in the form of an HTTP request. The UIW servlet receives the information coming from a user interface and decodes it.
- *Create an XML Message:* The information received from the UI is embedded in a XML message, which is consequently sent out to the corresponding application or DHE module for processing. Necessary information, such as Request ID, destination module ID, Message Type, etc., should be included as part of the header of the XML message.
- *Send an XML Message:* After an XML message is generated, the UIW has to send the message to the proper application or engine module. The messages are sent using the methods provided by the Message Manager.

The Traversal Path Manager inside the Message Manager routs the messages to their destinations.

- *Receive an XML Message from the Engine:* After sending a message, a thread will be waiting for a response from the application or module. This UIW will receive the XML messages from DHE using the method provided by the Message Manager.
- *Extract Content of an XML Message:* The UIW extracts the information to be displayed from the output XML messages. This messages include the output document, a list of elements-of-interest, meta-data, etc.
- *Format the Document to be Displayed in the User Interface:* The output document should be formatted to comply with the requirements of the user interface. Furthermore, the UIW also needs to take account of the additional features provided by DHE: frames, anchors, metadata, etc.
- *Send the Document to the User Interface:* This UIW is responsible of sending the output document for display in the user interface. This document should be in browser understandable format for the browser to be able to render it.

4.2.4 Dynamically Mapped Information System Wrappers

Dynamically Mapped Information System Wrappers (DMISW) must handle the communication between the application and the Hypertext Engine. Either the DMIS or the DMIS wrapper must provide the following support to integrate with the DHE. This list closely resembles compliance sets identified by OHS researchers [Davis 94].

- *Communicate with the DMIS:* The DHE must intercept the communications between the DMIS and its user interface. DMIS messages are routed through the DMIS wrapper to the DHE. The engine then can map hypermedia link anchors to it. The more loosely coupled (the more independent the computational and interface portions) and modular an information system is, the simpler the hypermedia integration will be. This also includes the processing of the messages coming from the UI to comply with the DMIS' API.
- *Message markup:* The DHE must identify DMIS objects in messages in order to determine whether to map hypermedia anchors to each. Currently the DHE relies on the DMISW to identify and mark up each DMIS element in some way, so the DHE knows where it lies in the message and what kind of object it is. This limitation could be alleviated for object-oriented DMISs if the DHE can query the DMIS objects embedded in messages. Otherwise the DHE would have to rely on some kind of sophisticated content analysis. An advanced DMISW could employ, for example, sophisticated lexical analysis techniques to infer undeclared semantics. The DMIS also could provide some content analysis routines for interpreting its messages.
- *Unique IDs:* Because the DHE maps contingent annotations and other links to DMIS element identifiers, the DMIS needs unique identifiers on each object the DHE could turn into a link anchor. This includes DMIS documents. In the current implementation the identifiers are coded as Uniform Resource Identifiers (URI).

- *Application developers must supply bridge laws:* The person who knows the DMIS best, the system designer who builds or maintains it, should provide its bridge laws. The information system builder must be both willing to and capable of developing a set of bridge laws that accurately captures the structure of his system. The DMIS must have a robust, inferable structure in order for the builder to capture it in bridge laws. Once in place, the bridge laws should map a hypermedia network to any of the system's application document instances, worksheets, program, etc. (Instance builders and users need have no knowledge of bridge laws. To them, hypermedia functionality occurs automatically).
- *Generalized Bridge Laws:* Each application developer must provide his own set of bridge laws. Nevertheless, it is possible to collect bridge law libraries that would map classes of information systems. This generalized or 'standard' bridge law sets could handle the models, attributes, data and operations found, e.g., in relational database applications, spreadsheet packages, or rule-based expert system shells. The builder of, say, a new database application would only have to match the elements in his system to those in the standard database set. The standard set would provide most of the bridge laws for his system. This would reduce the builder's effort both in determining which kinds of bridge laws would represent his system adequately and in developing these laws.
- *Other hypertext functionality:* Even if a DMIS has no bridge laws and passes messages without objects, the hypermedia engine still will provide other

standard hypermedia functionality (user annotation, backtracking, etc.) In this case the user will not be able to access DMIS items or operations in a hypermedia fashion through the DHE.

4.2.4.1 Relational Database Wrapper. The Relational Database Wrapper Module (RDWM) is an example of a particular DMIS wrapper. It acts as the "wrapper" around a Relational Database Management System (RDBMS). This enables the DHE to provide hypermedia support to it. The RDWM receives requests from other modules via the Gateway for data and/or meta-data from the RDBMS. The RDWM translates these requests into ANSI-92 SQL, which is the native messaging format for the RDBMS. Depending on the nature of the request, it then executes a SQL command, translates the result into a Gateway message, and asks the Gateway to route it to its final destination (the source of the original request). The RDWM works for all 'instances' of the DMIS, that is any relational database stored in it.

Functionality Overview

- Register itself with the Gateway.
- Receive requests (in the form of an XML message) from other Modules/Engine Components via the Gateway.
- Extract SQL Command from the request.
- Execute SQL Command.
- Decide which metadata to retrieve.
- Retrieve required metadata from the RDBMS.
- Retrieve RDBMS instance information.

- Mark elements with Bridge Laws and/or metadata.
- Create User Interface (UI) to display results.
- Return Response (formatted as an XML message).
- Create UI to interact with the RDBMS.

4.2.4.2 Spreadsheet Wrapper. The DHE Project currently provides different applications the ability to dynamically generate links on a user interface system. One of the DMIS within the current environment is the spreadsheet application. This module is a wrapper for this kind of DMIS, making it the Spreadsheet Wrapper Module (SSWM).

The spreadsheet Wrapper is designed so that users working with any spreadsheet can get automated linking and hypertext navigation features. The users are also shown the underlying spreadsheet model.

The SSWM builds three documents: the spreadsheet itself in HTML form, the spreadsheet model or schema, and the assumption values. The model describes the underlying schema for a cell of type formula. It is obtained by an external system, which outputs the schema of a spreadsheet when a spreadsheet is given as input. The assumption values in the spreadsheet are basically all the constants in the spreadsheet.

One of the objectives of this module is to enable the user to perceive the underlying logical schema of a spreadsheet without any user input. This will create a better understanding of the interactions between various elements of a spreadsheet. The elements in this case could be cells belonging to different tables in different sheets within a single workbook file. The user only needs to execute a macro on top of his/her workbook to obtain the description of different elements within a workbook. A text file containing the schema for each sheet is created within the same directory in which the application resides. A standard naming convention has been used to enable the user identify the different schema files generated for his / her workbook.

Functionality Overview

In brief, the SSWM provides automated linking and hypertext navigational features to the spreadsheet and makes the underlying model of the spreadsheet explicit to the user.

The various functions of the wrapper are:

- Register with Gateway.
- Send and receive messages.
- Display the necessary form in the UI for the user to enter the Spreadsheet name.
- Show Spreadsheet: Building document to display the Spreadsheet, its schema and the assumption values of the spreadsheet in individual frames in the browser using the UIW.
- Identify Elements: Identify all possible elements of the spreadsheet and assign element Ids to them.
- Generating Metadata: For each identified element of the document, relevant metadata should be generated and passed back to BLEM for it to add relevant links. The SSWM should send this Metadata only on demand.

- Provide Bridge Laws: Provide Bridge laws to BLEM for this module to generate links.
- To execute the spreadsheet and any SSWM commands underlying the links.

4.2.5 Menu Manager

The main objective of the Menu Manager module is to provide menu information of the application dynamically to the UIW module. UIW divides the HTML page into four frames viz., Main, Menu, Meta and Link (See Figure 4.2). The Menu frame is used to display menu items related to the underlying applications. This information is obtained by querying the Menu Manager.

Menu manager keeps all application's complete menu information in a database. The menu manager can update this database whenever it receives new menu information from the application wrappers.

Functionality Overview

- Register with Gateway.
- Accept messages from application Wrappers and extract menu Information from request/message.
- Add or update menu manager's database depending on message.
- Receive request from BLEM via Gateway.
- Retrieve necessary information from the database .
- Create DisplayDoc message or update DisplayDoc message send by application wrapper and send it to gateway.
- Identify different element types for this module and provide Bridge laws to BLEM for them to generate links.

4.2.6 User Preferences

The basic role of this module is to interact with other modules in the system in order to gather from the users, maintain, and provide any user preferences that any of the modules may use. Those preferences may concern what is being sent to the user interface wrapper module, how certain modules should proceed in executing their tasks, and many others. The number and kind of user preferences any of the modules may desire to have is not known in advance; thus, user preferences have to be maintained dynamically based on what requests or notifications arrive at the UPM. The interaction between UPM and the rest of the modules is done with internal messages.

The User Preferences Module manages information about users and user preferences. The UPM provides user management procedures like user log in and log off, and changing the user's own password. In addition, the UPM maintains information on specific user classes within user groups in the system.

User classes dictate privileges and preferences that are associated with a particular user. Users who belong to the class of administrators are given additional functionality. Administrators can add new users and remove current users from the system.

In terms of user preference management, UPM receives new preferences from other modules and stores them in its database. At any time, a module can request to retrieve a value of a user preference, have its values changed, or have it removed. Preference values are to be changed mainly by the users. Users make changes to their preferences through HTML forms. Again, administrators have additional functionality in this case. They can set preference values for any user in the system. They can also adjust default values for a given user class or set preference values for all current users of a chosen user class.

Functionality Overview

The User Preferences Module, as the name implies, manages user preferences. Specific user preferences are defined by modules in the engine, and are submitted to UPM for maintenance. Preferences can be used to describe user settings for various objects. The UPM itself is not concerned with the meaning of each of the preferences. It provides mechanism for storing, retrieving and changing their values for each of the users either by a module of the system or by a user

UPM is allowed to interact with UIW and any other module. Communication with modules and components of the engine is done through XML messages. Large part of communication is done with UIW because of many operations provided by UPM which are done by users. UPM generates and sends HTML forms where users enter information or make their choices. The forms include forms for log in, UPM menu, changing password, adding new users and removing users, forms for selecting modules and users to set preferences, and forms with preference values.

User preferences can be used in the engine for many tasks. They can be used to define many user settings. For instance, users can choose how they want certain information displayed, what kind of information they want to see, in what way a module should process information for them, etc. They can be also combined with bridge laws to provide specific conditions for generation of links which can be adjusted by users. In that sense, they directly relate to the goal of the and hypermedia engine project and are a useful tool in structuring the information.

As mentioned earlier, meaning and purpose of the preferences is not a concern of the UPM. Therefore, there is no restriction on what the preferences could be and what they refer to as long as they are defined in the proper format.

4.2.7 Database Schemas

The purpose of this module is to provide schematic information for database queries. This enables DHE to provide domain-specific link in a generic way for any database. There are many instances where the user views the end result of a query which are tuples generated by the query without having any understanding of the underlying schema of the database from which the query results were retrieved. Many applications run database queries without giving the user any direct access to schematic information. The DB Schema module provides the ability to show to the user or other interested parties the schematic information of a database from which the query was retrieved. The user has the ability for find where each value shown in a query result resides in the Entity Relationship schema of the database as well as the Physical Schema of the database. The user then has the ability to find the values in other tables and entities which are related to that particular value.

The database schema module runs in conjunction with the RDBWM module to provide a hypermedia wrapper around Relational Database

Management System. The function of the RDBWM module is to show metadata for values retrieved from the database. Most of the meta-data provided by the RDBWM module can be obtained through the ODBC or JDBC interface APIs for the particular RDMS. This type of metadata may include the name of the table from which a certain value was retrieved, if the value is a primary or foreign key, the data type of the value and so forth.

The DB Schema module adds to the metadata already provided by the RDBWM module by providing the ability to show schematic information for the values retrieved from a database query. The schematic information about a particular database has to be entered through a user interface by a system developer or administrator. When a query is issued by a user or application, the DB Schema module checks its internal database to find if the schematic information is stored for the database for which the query is issued. If schematic is stored for the query then it adds links to query result so that the user has the ability to view the underlying E-R schema as well as the physical schema of the database from which the query result was retrieved.

The DB Schema module creates three frames for the user when the query is issued. One frame shows the entity relationship schema of the database from which the query was issued. The second frame shows the Physical Schema of the database and the third shows the query result or the values retrieved for a particular database query.

Functionality Overview

The database schema module enables hypermedia functionality to dynamically generated documents retrieved from relational database management systems. It adds hypermedia links to elements embedded within results obtained from database queries allowing users to gain a better understanding of the internal structure of the data from which displayed information was generated.

The database schema module stores the schema information about a database entered by a system developer or administrator in its own internal database. When a database query is issued within the hypermedia engine structure, the module creates two additional frames in the browser where the query results are displayed. One frame displays the entity relationship diagram and the other physical or relational schema of the queried database in addition to frame displaying the query result. Each element, like the column names and the values in this context, displayed in the query result frame is turned into an anchor with several links. These links show to the user the structure of the database from which the query result was obtained by highlighting elements within the entity relationship schema frame or the physical schema frame. The elements within the entity relationship frame are entity, attribute and relationship. The elements within the physical schema frame are table, column and constraint.

The database schema module works along with the database RDBWM module to provide a complete set of hypermedia functionality to relational database management system. The RDBWM module creates its own set of links to elements received from database query. These links expose information available to the programmer but not usually displayed to the user about the database such as Number of Tables, Database Host Name, Database URL, Column Name, Number of Columns In This Result, Precision and so forth. The database schema module adds additional element types to the types generated by the RDBWM module to provide schematic information about the database from which the query results are obtained.

The requirements the DHE project includes the development of a second DMISW for a Relational Database Management System that automatically generates links for a RDMS based on database's schemata. The user should be able to select an anchor representing a database object such as value, tuple name, field name or table name and the module should then figure out all possible links based on the database's two schemata: the original E-R diagram (or a schema close to it), and the 3rd normalized form. If the user follows on of these links, the database will display the related database object at the other end which could be related value, tuple, field or table.

This module is a second DMISW for a RDMS and can have its own set of functionality including a DMIS that displays the two schemata and any other function that a DMIS rather than the DMISW should do.

The schema module specify a set of generic schema bridge laws (mapping rules) that the bridge laws module (BLEM) uses to determine the list of links. These bridge laws take the object selected by the user as input, and return the list of possible database link endpoints.

For the implementation of the bridge laws, a representation of each schema (the E-R schema and the relational or physical schema) is stored in a database. The bridge laws actually contain an SQL query (or a set of SQL queries) that acts over the schema representation. The hypermedia engine will know what kind of database object the user selected. The engine's bridge law module would then use the object type to figure out which bridge laws apply for that type of database object. Each bridge law represents one possible schema relationship for its kind of database object. Each bridge law contains an SQL query, and instantiates that SQL query with the object ID of the object the user selects. If selected, the result of the SQL query will be to display the related database object at the other end as mentioned above.

Part of the requirements were to develop a representation of both kinds of schemata. For now it will be assumed that a system developer will have to enter the schema information by hand for both kinds of schemata. The bridge laws for this module will rely on the schema being entered in this format, i.e., the SQL queries embedded in the bridge law be structured based on this representation.

Each schema will have to be displayed as an HTML document, which will be text based for now. Eventually a graphical representation based on this text document will be developed.

4.2.8 Index Manager

The hypermedia field has a rich feature set including guided tours, recommended paths, annotation, information overviews, sophisticated backtracking, and so on.

The purpose of the Index Manager Modules is to assist the hypermedia engine to implement hypermedia functionalities like guided tours and indexes.

The hypermedia engine can create guided tours by adding an ordered list of documents or objects to the guided tour. This could be done while visiting such documents or objects, by adding such documents or objects at once typing their locations (url, directory, etc) in the guided tours, or by adding them from history list/ bookmarks.

Functionality Overview

The Index Manager provides the functionality of managing and storing the indexes for the guided tours and other modules who wants to create index of hypermedia for the future reference. The Index Manager resides with the hypermedia engine and will closely be associated with the guided tour module. The Index Manager module have the functionality of maintain the persistence of the index or the path of the tour and will be able to manipulate the tour from user inputs. The module will provide such functions as creating index, loading index, adding node to an index, deleting node from an index, get specific information about a node in an index and etc. The Index Manager module will include the two parts: First, The Index Manager Storage which is implemented with RDBM to ensure the reliability, scalability, and safety. It provides storage of all of the tours information and allow the access through the Index Manager Module. Second, The Index Manager Module implemented with Java to make it potable, maintainable, and reusable. The Index Manager Module communicates with other modules like Guided Tour through the Gateway of Hypermedia Engine based on an client-server architecture to provide efficient service. The Index Manager Module accesses the Index Manager Storage through the JDBC to increase the transparency of the program.

4.3 Web Browser User Interface

The user interface system and/or the user interface wrapper must provide the following support to integrate with the DHE.

- *Provide hypermedia prompts:* The interface must distinguish hypermedia anchors in some way from other window content. The interface must distinguish anchor selection from other user actions, thus providing a mechanism to invoke link traversal. If users can edit the hypermedia anchors, then the system must distinguish selection for editing from selection for traversal.
- *Menus:* The interface must display DHE and DMIS menus in some fashion, though this could be in some crude manner such as making each menu a hypermedia anchor, which would be placed in all documents.
- *Identifying objects in UI workspaces:* If users may edit UI documents, then the interface must track hypermedia anchors when their positions shift, so that users still can select and traverse them. Some research has been done to provide ways to find anchors once text has shifted. [Kacmar 93] notes the need for coupling a call to windowing systems with a call to an underlying interface component to obtain the object identifier.

- *Passing information to the DHE:* When the user selects a menu or selects an anchor for traversal, the interface must inform the DHE.
- *User dialogs:* The user interface must support some kind of forms or dialog box to collect user input. Applications sometimes require users to input parameter values, such as the value of a parameter, over which to execute a command.
- *Accept displays from the DHE:* In most information systems users create documents manually. With a hypermedia engine, the UI must accept the externally-generated documents with embedded information that the DHE passes. Ideally the UI will handle dynamic changes as well. The engine may add additional objects to open documents (e.g., when users create their own annotations on the front-end workspace). Dynamic updating may change the display value of anchors, such as displaying the current price of a stock. This requires the UI to accept external interrupts so it can accept information not caused by (or anticipated in response to) a user interface event.

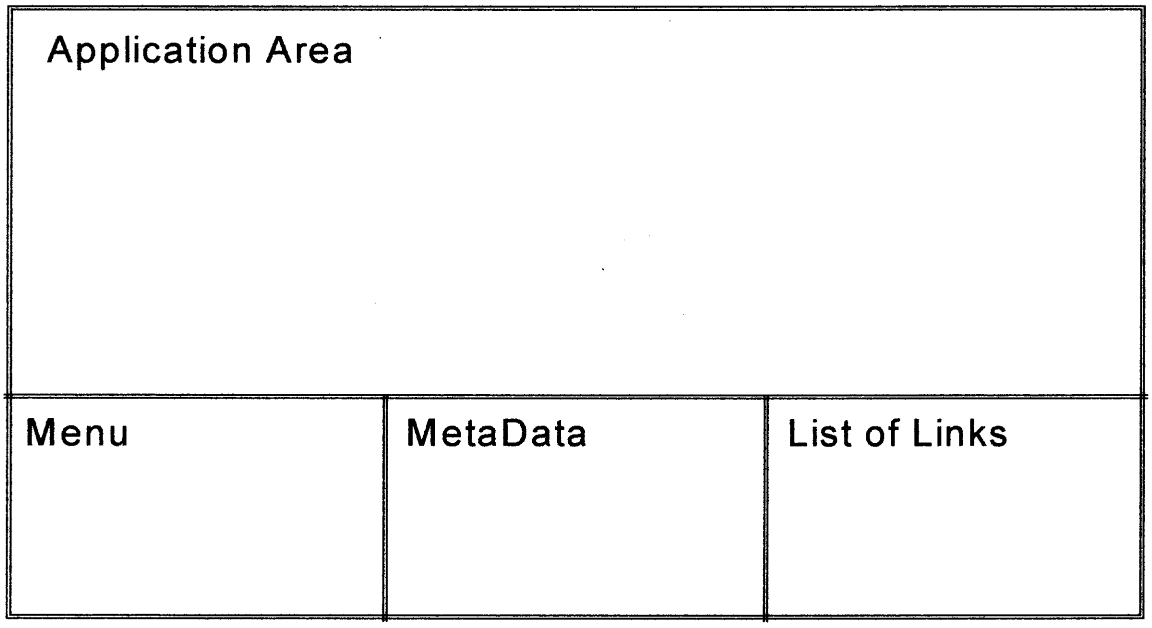


Figure 4.2 DHE 1.0 Web Browser User Interface

4.4 Operation Overview

Hypermedia functionality and interrelationship access supplement the application's normal operations. In terms of information representations, DHE need to model hypermedia data structures, hypermedia navigation structures, and if not otherwise available, application metainformation and the application internal structure (i.e., its design or schema). All are critical for automated dynamic mapping.

In keeping with the goal of altering applications as little as possible, DHE do not alter the application's existing data or access structures, but it must be able to interact with them. The mapping, in turn, is critical for retrieving information, i.e., user access. DHE builds hypermedia functionality on top of the hypermedia data structures using this mapping. The richer the data representation, the more detailed the mapping. This, in turn, affects both the types of navigation available and the types of interrelationships that can be inferred automatically.

The engine uses a relatively simple hypermedia data model consisting of nodes, links and link markers-buttons or anchors. Each carries a rich and flexible set of attributes for capturing the semantic and behavioral aspects of the application elements it represents. Bridge laws specify the actual mapping between each type of application element and its equivalent hypermedia construct. Bridge laws are logical rules for mapping two independent representational domains [Kimbrough 77], [Bieber 92], [Bieber 94], case model management and hypermedia in this case.

Mapping and retrieval occur as follows (See Figure 4.3). DHE intercepts all messages passing from the computational portion of an application to the interface, and uses bridge laws to map each appropriate element of the message to a hypermedia node, link or anchor. Users interact with these hypermedia representations on the screen. When the user selects a normal application command the hypermedia engine passes the command on to the application for processing. When the user selects a hypermedia engine command (such as a menu command to create an annotation), the hypermedia engine processes it entirely. When the user selects a supplemental schema, process, operation, structural, descriptive, information or occurrence relationship, the engine infers the appropriate application commands, meta-application operations (e.g., at the operating systems level or schema level) or hypermedia engine operations that will produce the desired information. When the user selects a contingent (ad hoc) relationship, the hypermedia engine retrieves the annotation. The interface also handles all other interaction with the DMIS application, such as form-filling and directly manipulating of DMIS objects. Thus users access the entire functionality of the DMIS as well as the supplemental hypermedia features through the independent front-end interface system.

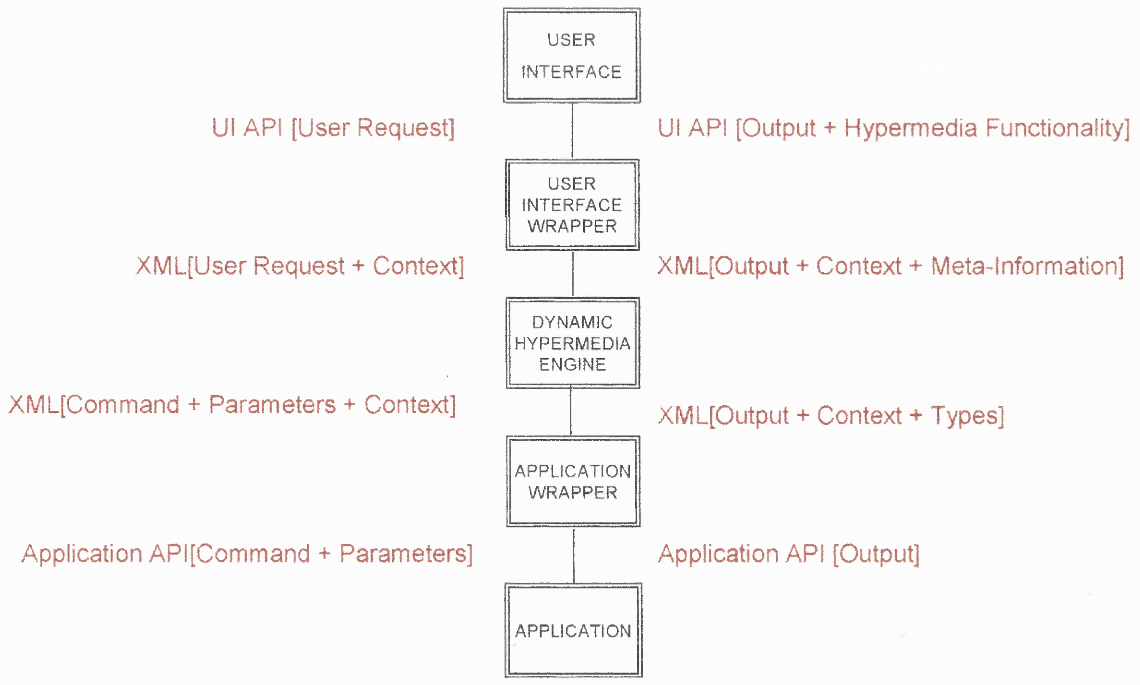

USER INTERFACE
UI API [User Request]
UI API [Output + Hypermedia Functionality]
USER INTERFACE WRAPPER
XML[User Request + Context]
XML[Output + Context + Meta-Information]
DYNAMIC HYPERMEDIA ENGINE
XML[Command + Parameters + Context]
XML[Output + Context + Types]
APPLICATION WRAPPER
Application API[Command + Parameters]
Application API [Output]
APPLICATION

Figure 4.3 DHE Operation

DHE is capable of working with applications that can use an independent front-end interface system, such as those which provide an application programming interface (API). Since DHE needs to use a front-end that can display selectable hypermedia link markers, World Wide Web browsers qualify, and the current implementation of DHE includes the capability of interacting with user interfaces on the Web.

4.5 Summary

DHE automatically generates links at run-time. DHE also constructs more sophisticated navigation techniques on top of these links. The metadata, links and navigation, as well as annotation features, supplement the application's primary functionality.

DHE runs in parallel with other applications and dynamically provides hypermedia functionality. There is a run-time mapping of the application information and relationships to hypertext objects

DHE intercepts the communication between the application's computational section and its interface section, automatically detecting the location of interrelationships in messages and documents based on the knowledge of the application's internal structure. Bridge Laws provide the mapping between the structural relationships of the application and the hypertext features.

To provide a DMIS application with hypermedia support the developer has to declare the contents of its DMIS wrapper and knowledge base.

The DHE 1.0 architecture consist of several modules:

- User Interfaces.
- User Interface Wrappers.
- Gateway.

- Traversal Path Manager.
- Log Manager.
- Bridge-Law Element Mapper.
- Link Bridge Laws.
- Dynamically Mapped Information System Wrappers.
- Dynamically Mapped Information Systems.

The user interface system and/or the user interface wrapper must provide the following support to integrate with the DHE: hypermedia prompts, menus, identify objects, pass information to the DHE, user dialogs, and accept displays from the DHE.

DHE intercepts all messages passing from the computational portion of an application to the interface, and uses bridge laws to map each appropriate element of the message to a hypermedia node, link or anchor.

When the user selects a normal application command the hypermedia engine passes the command on to the application for processing. When the user selects a hypermedia engine command, the hypermedia engine processes it entirely.

CHAPTER 5

DHE 1.0 IMPLEMENTATION

The DHE version 1.0 consists of several well-defined and separate processes, each possibly running on a different platform:

- *User Interface:* which usually runs on the user's computer. A web browser was selected as user interface for this prototype.
- *User Interface Wrapper:* A user interface wrapper has been provided to handle the communication between DHE and the user interface.
- *Gateway:* This module enables the communication between the Engine modules and works as the router for all the DHE internal messages. All Engine messages pass through the GW, which then redirects them to the appropriate module.
- *Bridge-Law Element Mapper:* stores and retrieves the bridge laws required to automatically generate links.
- *Relational Database Wrapper:* A wrapper for a relational database system.
- *Dynamically Mapped Information System:* The DMIS in this implementation is the Oracle Database Management System.
- *User Preferences:* Allow the specification of preferences based on group, group inheritances, or individual users. Any other module can store and manage their preferences.

5.1 Functionality

The DHE 1.0 provides the following functionality:

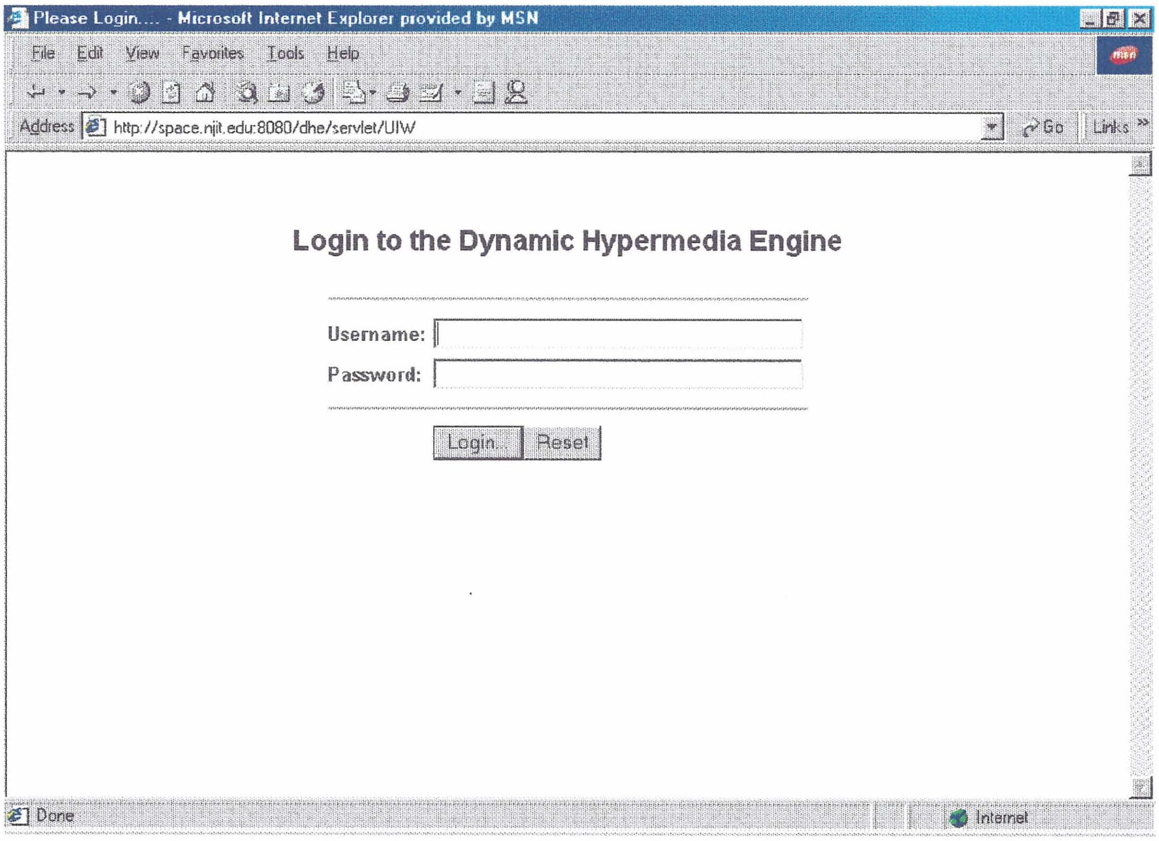
- Store element classes and mapping rules.
- Store users, user groups, and modules preferences.
- Accept input from the User Interface. In version 1.0 a User Interface Wrapper for Web Browsers has been implemented.
- Process requests and return the results to the User Interface.
- Identify the elements with relationships that appear in the results.
- Display the output along with the added meta-information: anchors, links, metadata, menus, etc.

5.2 Example of Use

To use the applications currently supported by the Dynamic Hypermedia Engine 1.0 it is necessary to visit the URL

http://space.njit.edu:8080/dhe

with a Web browser. This page will display the login screen. Users can enter their Username/Password combinations to start using DHE. Visitors can use the 'guest' Username (password 'guest') to access the Engine.


Please Login.... - Microsoft Internet Explorer provided by MSN
File Edit View Favorites Tools Help
Address http://space.njit.edu:8080/dhe/servlet/UIW
Go
Links
Done
Internet

Login to the Dynamic Hypermedia Engine

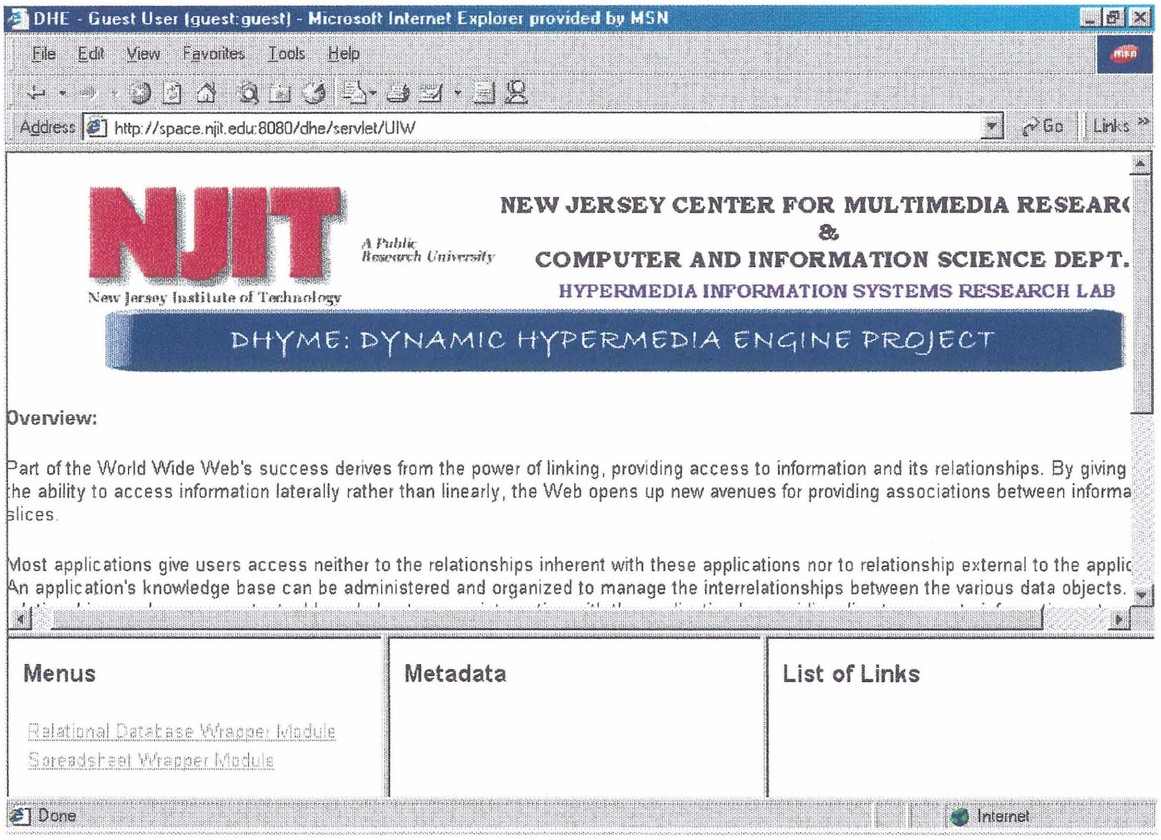
Username:

Password:

Login... Reset

Figure 5.1 DHE Login Screen

After Login DHE displays it Main screen with information about the project and showing the default layout described in section 4.3 . Figure 5.2 shows two applications available: The Relational Database Wrapper Module, and the Spreadsheet Wrapper Module.


DHE - Guest User (guest:guest) - Microsoft Internet Explorer provided by MSN
File Edit View Favorites Tools Help
Address http://space.njit.edu:8080/dhe/servlet/UIW
Go
Links
NJIT
A Public Research University
New Jersey Institute of Technology
Done
Internet

NEW JERSEY CENTER FOR MULTIMEDIA RESEAR(
&
COMPUTER AND INFORMATION SCIENCE DEPT.
HYPERMEDIA INFORMATION SYSTEMS RESEARCH LAB

DHYME: DYNAMIC HYPERMEDIA ENGINE PROJECT

Overview:

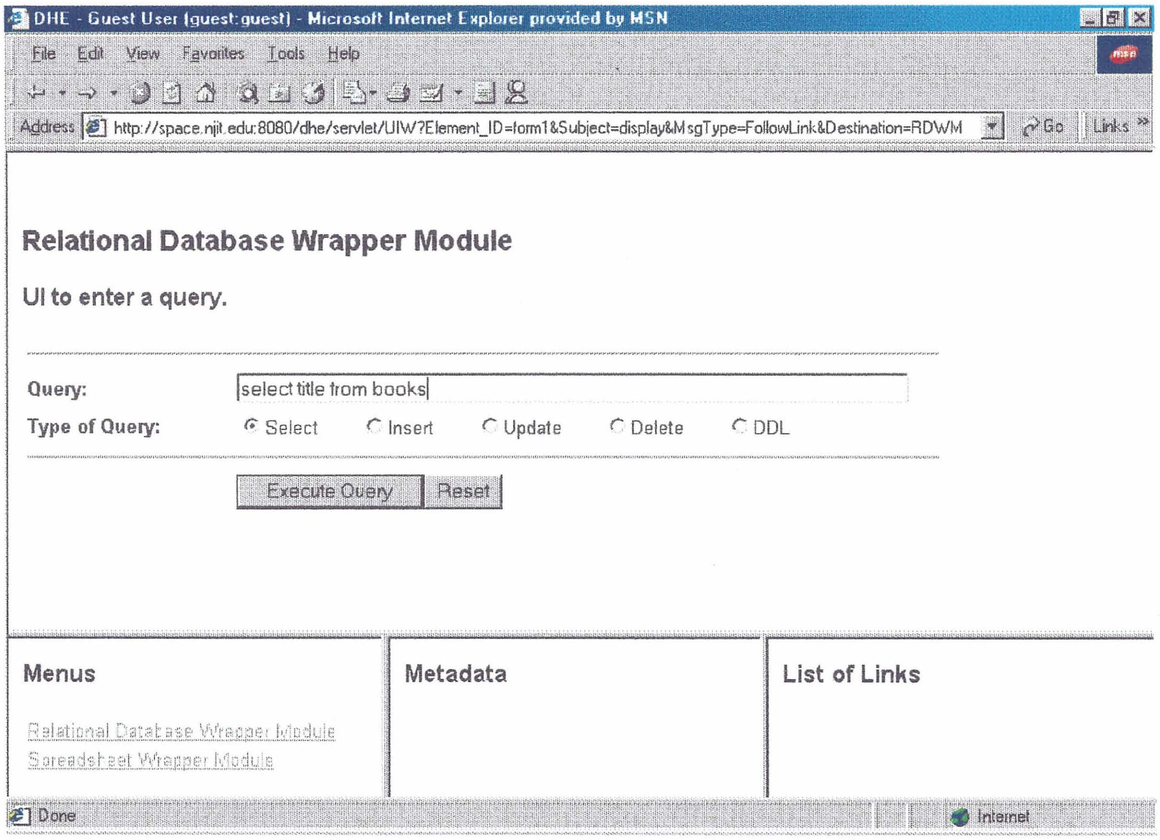
Part of the World Wide Web's success derives from the power of linking, providing access to information and its relationships. By giving the ability to access information laterally rather than linearly, the Web opens up new avenues for providing associations between informa slices.

Most applications give users access neither to the relationships inherent with these applications nor to relationship external to the applic An application's knowledge base can be administered and organized to manage the interrelationships between the various data objects.

Menus	Metadata	List of Links
Relational Database Wrapper Module Spreadsheet Wrapper Module		

Figure 5.2 DHE Main Screen

Users can select the desired application by clicking on its name in the screen. Selecting the 'Relational Database Wrapper Module' will invoke the application and it will send its initial screen to be displayed in the user interface.


DHE - Guest User (guest:guest) - Microsoft Internet Explorer provided by MSN
File Edit View Favorites Tools Help
Address http://space.njit.edu:8080/dhe/servlet/UIW?Element_ID=form1&Subject=display&MsgType=FollowLink&Destination=RDWM
Go
Links
Execute Query
Reset
Relational Database Wrapper Module
Spreadsheet Wrapper Module
Done
Internet

Relational Database Wrapper Module

UI to enter a query.

Query: select title from books

Type of Query: [✓ Select] Insert Update Delete DDL

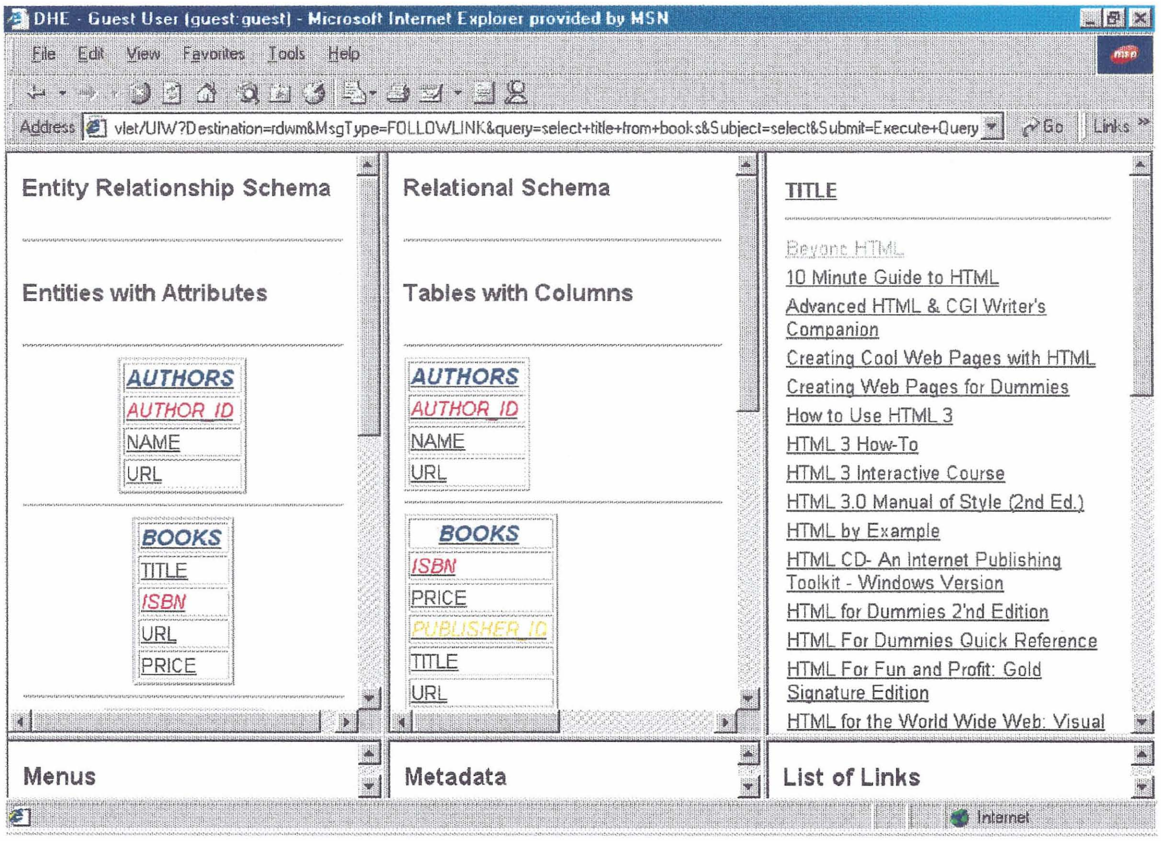
Menus

Metadata

List of Links

Figure 5.3 RDBWM Query Input Screen

In the Relational Database Wrapper application users can type SQL commands to be executed by a Database Management System, (Oracle in this case). The query in this example requests all the titles present in the table named ‘books’.


DHE - Guest User (guest:guest) - Microsoft Internet Explorer provided by MSN
File Edit View Favorites Tools Help
Address
Go
Links
Internet

vlet/UIW?Destination=rdwm&MsgType=FOLLOWLINK&query=select+title+from+books&Subject=select&Submit=Execute+Query

Entity Relationship Schema

Entities with Attributes

AUTHORS
AUTHOR_ID
NAME
URL

BOOKS
TITLE
ISBN
URL
PRICE

Relational Schema

Tables with Columns

AUTHORS
AUTHOR_ID
NAME
URL

BOOKS
ISBN
PRICE
PUBLISHER_ID
TITLE
URL

TITLE

Beyond HTML
10 Minute Guide to HTML
Advanced HTML & CGI Writer's Companion
Creating Cool Web Pages with HTML
Creating Web Pages for Dummies
How to Use HTML 3
HTML 3 How-To
HTML 3 Interactive Course
HTML 3.0 Manual of Style (2nd Ed.)
HTML by Example
HTML CD- An Internet Publishing Toolkit - Windows Version
HTML for Dummies 2'nd Edition
HTML For Dummies Quick Reference
HTML For Fun and Profit: Gold Signature Edition
HTML for the World Wide Web: Visual

Menus

Metadata

List of Links

Figure 5.4 DHE Display of Query Results

The result of the query is enhanced by the DHE. Instead of displaying a simple text-based table the Engine performs the following transformations:

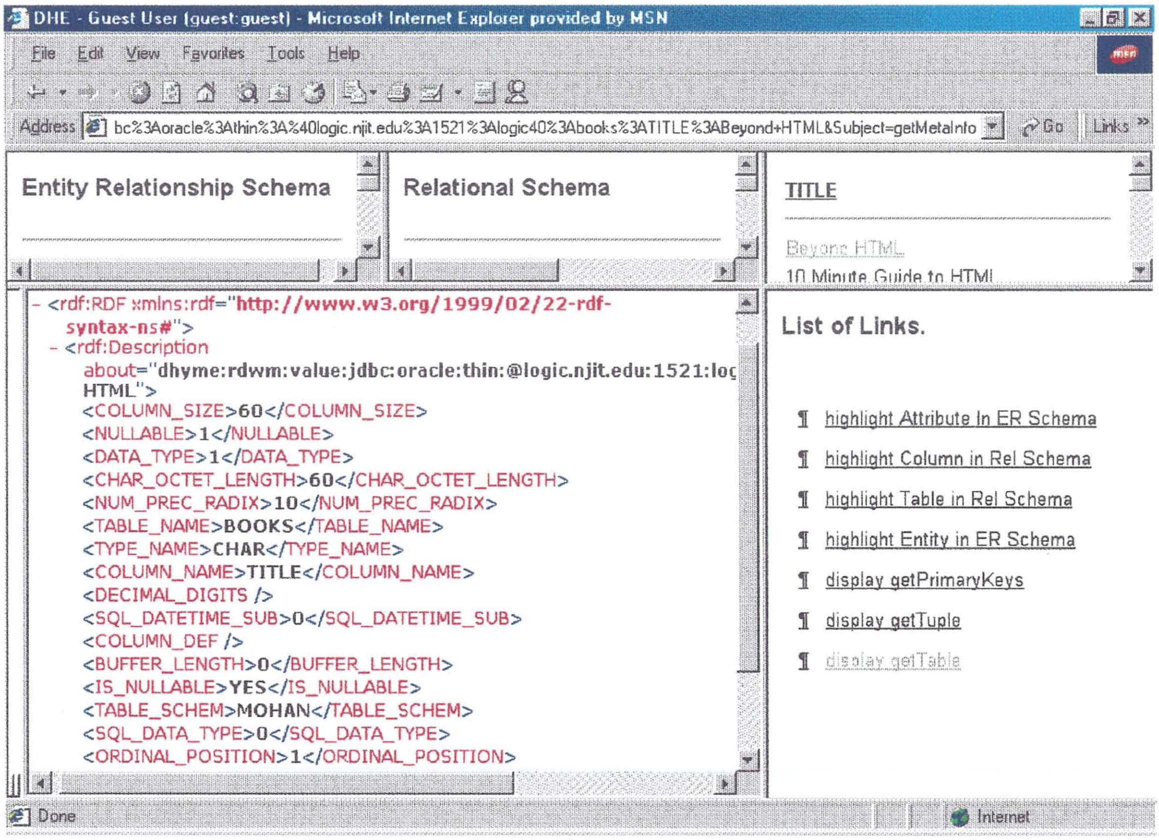
1. Translates the output of the DBMS to XML (RDBWM)

2. Identifies which elements present in the output have meta-Information (BLEM)

3. Includes the frames with the Entity-Relationship and Relational Schemas (DB Schema)

4. Assembles all the pieces of meta-Information and converts them to HTML (UIW)

The elements with meta-Information (i.e. links and/or metadata) are shown as anchors (underlined).


DHE - Guest User (guest:guest) - Microsoft Internet Explorer provided by MSN
File Edit View Favorites Tools Help
Address
Go
Links
Entity Relationship Schema
Relational Schema
TITLE
Beyond HTML
10 Minute Guide to HTML
List of Links.
Done
Internet

bc%3Aoracle%3Athin%3A%40logic.njit.edu%3A1521%3Alogic40%3Abooks%3ATITLE%3ABeyond+HTML&Subject=getMetaInfo

```
- <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
   syntax-ns#">
  - <rdf:Description
      about="dhyme:rdwm:value:jdbc:oracle:thin:@logic.njit.edu:1521:log
      HTML">
      <COLUMN_SIZE>60</COLUMN_SIZE>
      <NULLABLE>1</NULLABLE>
      <DATA_TYPE>1</DATA_TYPE>
      <CHAR_OCTET_LENGTH>60</CHAR_OCTET_LENGTH>
      <NUM_PREC_RADIX>10</NUM_PREC_RADIX>
      <TABLE_NAME>BOOKS</TABLE_NAME>
      <TYPE_NAME>CHAR</TYPE_NAME>
      <COLUMN_NAME>TITLE</COLUMN_NAME>
      <DECIMAL_DIGITS />
      <SQL_DATETIME_SUB>0</SQL_DATETIME_SUB>
      <COLUMN_DEF />
      <BUFFER_LENGTH>0</BUFFER_LENGTH>
      <IS_NULLABLE>YES</IS_NULLABLE>
      <TABLE_SCHEM>MOHAN</TABLE_SCHEM>
      <SQL_DATA_TYPE>0</SQL_DATA_TYPE>
      <ORDINAL_POSITION>1</ORDINAL_POSITION>
```

¶ highlight Attribute In ER Schema
¶ highlight Column in Rel Schema
¶ highlight Table in Rel Schema
¶ highlight Entity in ER Schema
¶ display getPrimaryKeys
¶ display getTuple
¶ display getTable

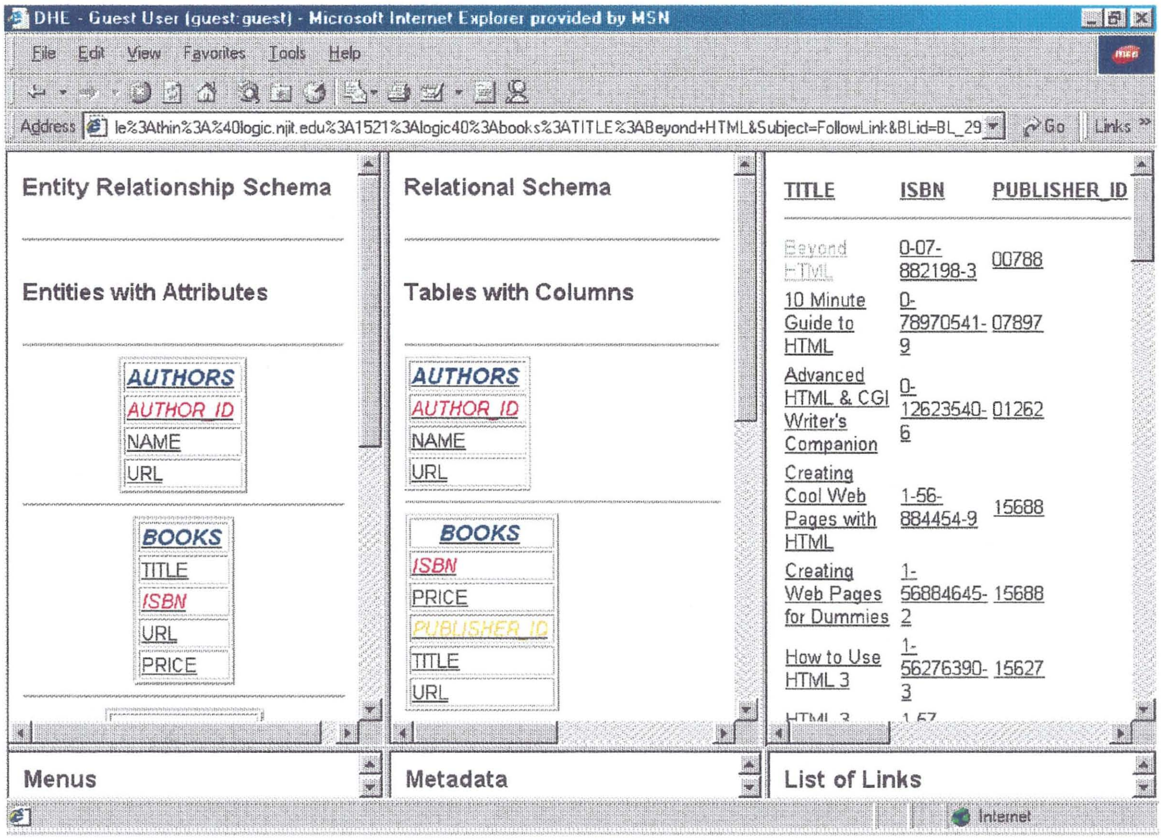
Figure 5.5 Meta-Information for element 'Beyond HTML'

The user can then proceed to select any of the elements-made-anchors to retrieve additional information about them. When the user clicks on an anchor DHE searches its tables to find relationships and metadata, and displays them in the appropriate frames on the user interface.

It is even possible to obtain meta-Information about the meta-Information being displayed. To get meta-Information about links in the list-of-links the user can select the icon ¶.

The links represent instructions for different DHE modules, or other applications, to produce or retrieve related information.

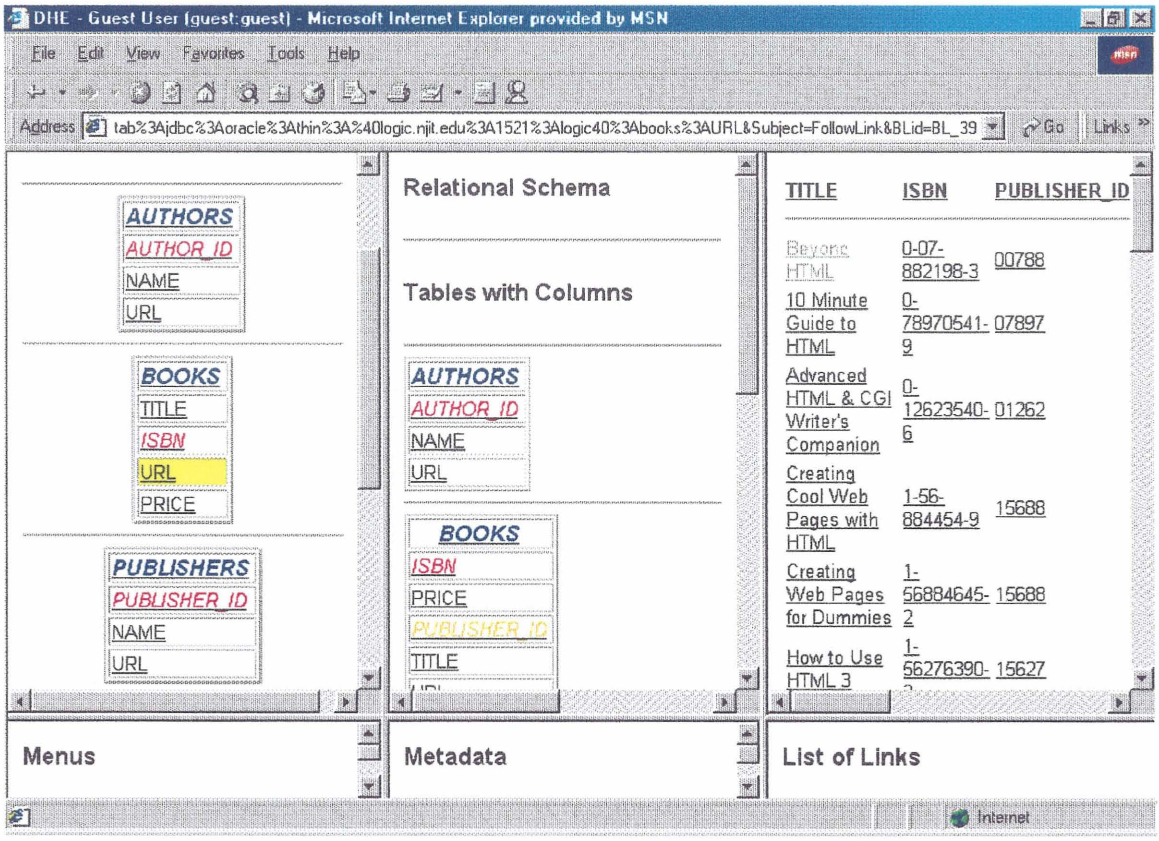
The result of selecting the 'display_getTable' link is shown in Figure 5.6 . DHE send the instruction to the corresponding module, RDBWM in this case, and the DHE cycle is repeated once again.


DHE - Guest User (guest:guest) - Microsoft Internet Explorer provided by MSN
File Edit View Favorites Tools Help
Address le%3Athin%3A%40logic.njit.edu%3A1521%3Alogic40%3Abooks%3ATITLE%3ABeyond+HTML&Subject=FollowLink&BLid=BL_29
Go
Links
Entity Relationship Schema
Entities with Attributes
AUTHORS
AUTHOR_ID
NAME
URL
BOOKS
TITLE
ISBN
URL
PRICE
Relational Schema
Tables with Columns
AUTHORS
AUTHOR_ID
NAME
URL
BOOKS
ISBN
PRICE
PUBLISHER_ID
TITLE
URL
Menus
Metadata
List of Links
Internet

TITLE	ISBN	PUBLISHER_ID
Beyond HTML	0-07-882198-3	00788
10 Minute Guide to HTML	0-78970541-9	07897
Advanced HTML & CGI Writer's Companion	0-12623540-6	01262
Creating Cool Web Pages with HTML	1-56-884454-9	15688
Creating Web Pages for Dummies	1-56884645-2	15688
How to Use HTML 3	1-56276390-3	15627
HTML 3	1-57	

Figure 5.6 Result of Selecting the 'display_getTable' Link

Figure 5.7 shows the result of selecting the 'highlight_Attribute_In_ER_Schema' link. The list of links can contain links offered by different modules and/or applications. In each of this last two actions the request is handled by a different module, RDBWM in the first case, and DB Schema in the second.


DHE - Guest User (guest:guest) - Microsoft Internet Explorer provided by MSN
File Edit View Favorites Tools Help
Address tab%3Ajdbc%3Aoracle%3Athin%3A%40logic.njit.edu%3A1521%3Alogic40%3Abooks%3AURL&Subject=FollowLink&BLid=BL_39
Go
Links
AUTHORS
AUTHOR_ID
NAME
URL
BOOKS
TITLE
ISBN
URL
PRICE
PUBLISHERS
PUBLISHER_ID
NAME
URL
Relational Schema
Tables with Columns
AUTHORS
AUTHOR_ID
NAME
URL
BOOKS
ISBN
PRICE
PUBLISHER_ID
TITLE
Menus
Metadata
List of Links
Internet

TITLE	ISBN	PUBLISHER_ID
Beyond HTML	0-07-882198-3	00788
10 Minute Guide to HTML	0-78970541-9	07897
Advanced HTML & CGI Writer's Companion	0-12623540-6	01262
Creating Cool Web Pages with HTML	1-56-884454-9	15688
Creating Web Pages for Dummies	1-56884645-2	15688
How to Use HTML 3	1-56276390-	15627

Figure 5.7 Result of Selecting the 'highlight_Attribute_in_ER_Schema' Link

5.3 Review

The current prototype is operational. After working with DHE for some time some drawbacks have been detected:

- The main problem identified during the operation of DHE 1.0 is related to the efficiency of the system more than to the effectiveness.
- Since DHE has to perform several operations, access several tables stored in a remote database, and transmit and receive multiple messages through

the network, in addition to return the results to the user interface, the response time is higher than what would take a 'regular' application to fulfill the same request.

- Even though DHE's modules use JDBC (Java's ODBC implementation) to interact with the Database Management System (Oracle in this prototype), the implementation requires such DBMS to be present to be able to run, which limits the portability.
- The implementation of DHE 1.0 is somewhat biased towards the use of a Web browser as the user interface.
- The current DHE user interface is rudimentary and does not give a full sense of what it is possible to do.
- There is not an appropriate mechanism to handle error conditions during the processing of a request by DHE.
- A mechanism to monitor the execution and facilitate the debugging of the DHE, particularly during the integration of new modules and/or features.

5.4 Summary

The DHE version 1.0 consists of the following modules: User Interface, User Interface Wrapper, Gateway, Bridge-Law Element Mapper, Relational Database Wrapper, Dynamically Mapped Information System (Oracle Database Management System), User Preferences, and Database Schemas.

The DHE 1.0 provides the following functionality: store element classes and mapping rules; store users, user groups, and modules preferences; accept input from the User Interface; process requests and return the results to the User Interface; identify the elements with relationships that appear in the results; display the output along with the added meta-information.

A full example of DHE's operation shows the different output screens generated during the execution of the relational database system. The figures display the four frames defined by DHE, and the meta-information displayed in each one of them based on the user's input.

The prototype is operational, but some problems have been detected: performance (i.e. response time), dependency on database, implementation biased towards a Web Browser, rudimentary user interface, lack of error recovery and debug.

CHAPTER 6

DHE NEXT GENERATION (DHE NG)

After the completion of version 1.0 of the DHE, an updated DHE architecture is proposed. This enhanced architecture aims to correct some of the aforementioned problems as well as include the infrastructure required to implement new features. The main goals of the new design are:

- *Improve performance:* This is the main problem to solve since the response time of applications going through DHE is too high. There are several approaches that can be taken to offer a better response:
  - *Make DHE's core modules more tightly coupled.* Currently all DHE modules are distributed, each could be running in a different system as long as they have an Internet connection. This distributed model causes extremely communications overhead within the Engine since all message interchange between modules involves serializing objects, transmission over the network, and re-parsing. After working with the prototype it became evident that makes more sense to perform most of the core tasks 'locally'.
  - *Store information locally.* Another source of delay is due to the fact that all DHE modules store and retrieve information from a remote RDBMS, Oracle in this case. This of course has a significant time cost, among other logistical problems, and it also limits the potential portability of the Engine since it cannot run unless there is a RDBMS available. The

proposed solution uses XML files, instead of database tables, to store the information modules need.

    - *Maximize the metainformation added per request.* DHE 1.0 waits for the user to click on an anchor to submit a search for the list of links corresponding to that element type. This implies the need to send more messages and access more database tables (as mentioned in the previous points). The solution proposes to gather as much information as possible while returning the original user request.

- *Include additional Hypermedia functionalities:* This design should provide a solid conceptual architecture that would allow the inclusion of additional hypermedia functionalities:
    - *Guided Tours.* Guided tours provide the users the option to create or follow a path of documents or objects. The Guided Tour module should be able to maintain the persistence of nodes on the path of the tour and should be able to manipulate the tour from user inputs. When executing an index or Guided tours, the index module should receive the UI requests, not the applications. The IGT module will redirect the request to the appropriated module.
    - *Manual Linking.* This functionality enables the users to manually create new links between objects. These links should be stored in the Engine and be displayed when the document is revisited or regenerated. The new links should also be made available to other users or groups of

users. Users can opt to make a link bi-directional and must provide semantic types for each link and direction.

- *Annotations.* Offers the opportunity to manually create or retrieve annotations and link them to objects. The annotations have to be stored in the Engine and be displayed when the document is revisited or regenerated. The annotation may also be made available to other users or groups of users.

• *Support Features:* To be able to implement the hypermedia functionalities listed above, it is necessary to incorporate the following support features:

  - *Regeneration and Saving.* Some hypermedia features (like Guided Tours) require the ability to store, regenerate, and retrieve documents. Regeneration implies that the document to be displayed should be re-created from its original source. Saving means that a 'static' copy of the document is to be stored and, subsequently, retrieved without having to ask an application to reproduce it.
  - *Object Types and Identifiers.* Several objects, like applications, documents, and elements, need to be typed and uniquely identified to enable advanced hypermedia functionalities, like those that were mentioned above. DHE should define a strategy for the generation of identifiers. (Using XPath - XPointer)
  - *Target Areas.* DHE needs to allow users to define 'target areas' that will serve as points of origin or destinations for links, annotations, or

other kind of operations. These target areas could overlap with each other without loss of functionality. (Using XPath - XPointer)

- *Conditions.* There exist some structures inside DHE, like bridge laws, for example, which could be activated or not, depending on the current context. DHE should provide a mechanism to declare and evaluate such conditions.
- *Error recovery and debugging:* DHE requires the implementation of an error handling policy, as well as features that would facilitate the debugging and monitoring of the Engine processes.
- *DHE Administrative Applications.* It is possible to regard DHE as a whole like an application and create a interface which could allow access to the information stored in DHE's files for the system administrator or developers.

6.1 DHE NG Architecture

The basic DHE NG architecture remains very similar to DHE 1.0 at the System Level, with the Engine as middleware between the applications and the user interface. The idea of using 'wrappers' to habilitate the interaction with applications and user interfaces is maintained.

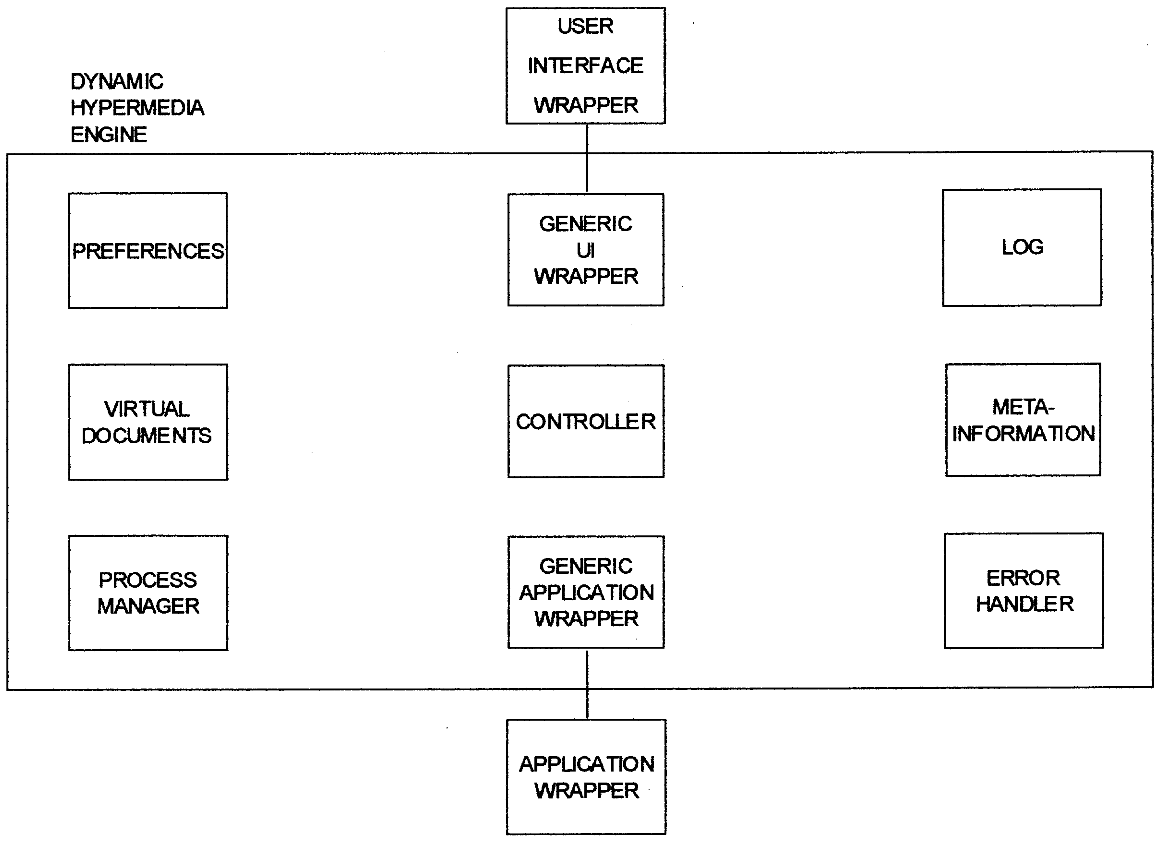


Figure 6.1 DHE NG Main Components

- *User Interface* (UI), it can be any kind of device that would allow the user to interact with the application.
- *User Interface Wrapper* (UIW), enables the communication between the Controller and the user interface. The UIW is responsible of handling several important functions: receives the users inputs (from the UI), creates the corresponding message objects for processing by DHE and delivers them to the Controller, manages user sessions, transforms the applications output to the appropriate format the UI recognizes, caches the metainformation returned by DHE, and serves the metainformation to the UI whenever it is possible.

- *Controller* (CTR), manages the processing of the message objects by invoking the appropriate modules.
- *Application Wrapper* (AW), identifies and marks the elements within the application's output for which relationships and/or metadata exist, manages the communication between the DHE and the application system, translates the user requests from the message objects to the application API, receives the output from the application and includes it in a message object.
- *Application* (AP), an application system that dynamically generates virtual documents from users requests.

6.1.1 Controller

The Controller (CTR), manages the execution of all the core components of the engine. As opposed to DHE 1.0, the Controller is not merely a message passing system. In DHE NG the Controller will instantiate and invoke several sub-systems that will provide the required functionality without having to transmit messages over the network. The Controller will utilize the following sub-modules:

Process Manager (PM)

The Process Manager stores the name and the order of the components that need to participate in the processing a message object.

- All DHE-VirtualDocuments go from the Source to the Processor and Back to the Source.

- All requests must produce and answer.
- Processors (i.e. modules) should update the Path before returning the VD.

Meta-Information Module (MIM)

The Meta-Information Module stores the mapping rules and adds the list of links to the message objects.

Meta-Information Scope Hierarchy (MISH)

The DHE's MISH is formed from the decomposition of the applications into addressable components at different levels. Within DHE, it is possible to define relationships as belonging to a particular level in the hierarchy and, by inheritance, to all levels below. The MISH consists of the following typical levels:

1. Engine Level. All output from DHE (from any application) will include relationships and metadata defined at this level. This includes the extra functionality provided by DHE to every application (like the creation of annotations and manual links, for example).
2. Application Level. All output coming from this particular application will include relationships and metadata defined at this level. This level could be used to define static application menus which are always present in any sub-part of the application.
3. Sub-Modules/Sections Level(s). Applications can be further decomposed in as many levels and sections as being convenient, defining relationships and metadata appropriate for each one of them.

4. Document Level. This level will allow applications to define meta-Information for all documents of a certain class.
5. Element Level. This is the original level used for elements in DHE 1.0 .
6. Lexical Level. It should be possible to store meta-Information for pre-determined lexical values. The lexical value may be, or not, an instance of an element.
7. Instance Level. This specifies meta-Information for particular occurrences of documents or types. It is necessary to store all the information required to uniquely identify the object.

Meta-Information Scope Hierarchy Rules and Constraints

- The Engine Level is the root of the hierarchy. The 'DHE' type is, currently, the only value defined for this level.
- All types in the hierarchy must have at least one parent. Types could have more than one parent. The Engine Level type, 'DHE', is the root of the hierarchy and is the only type allowed in the hierarchy without a parent.
- Levels and types can only have parents which are in the same or higher scope levels. Circular hierarchies should be detected and prevented.
- If an application is divided further in sections, the sections sub-hierarchy must have the application level type as root.
- The Application Hierarchy implies inheritance from the higher levels to the lowest. This means that children will have all the meta-Information of their ancestors. The list of relationships of an instance, for example, will also

include the relationships of its type, its document, its section(s), its application, and the DHE.

- Element, and lexical types can have parents at any other level

The MISH builds a graph that can be traversed using the W3C XPath specification. This becomes the basis for types identifiers:

DHE[[::APPLICATION:application[::SECTION:section]*][::DOCUMENT:documentType [:instance]][::ELEMENT:elementType[:instance][::LEXICAL:lexical]]]

The DHE-MISH is formed with the main components shown in Figure 6.2:

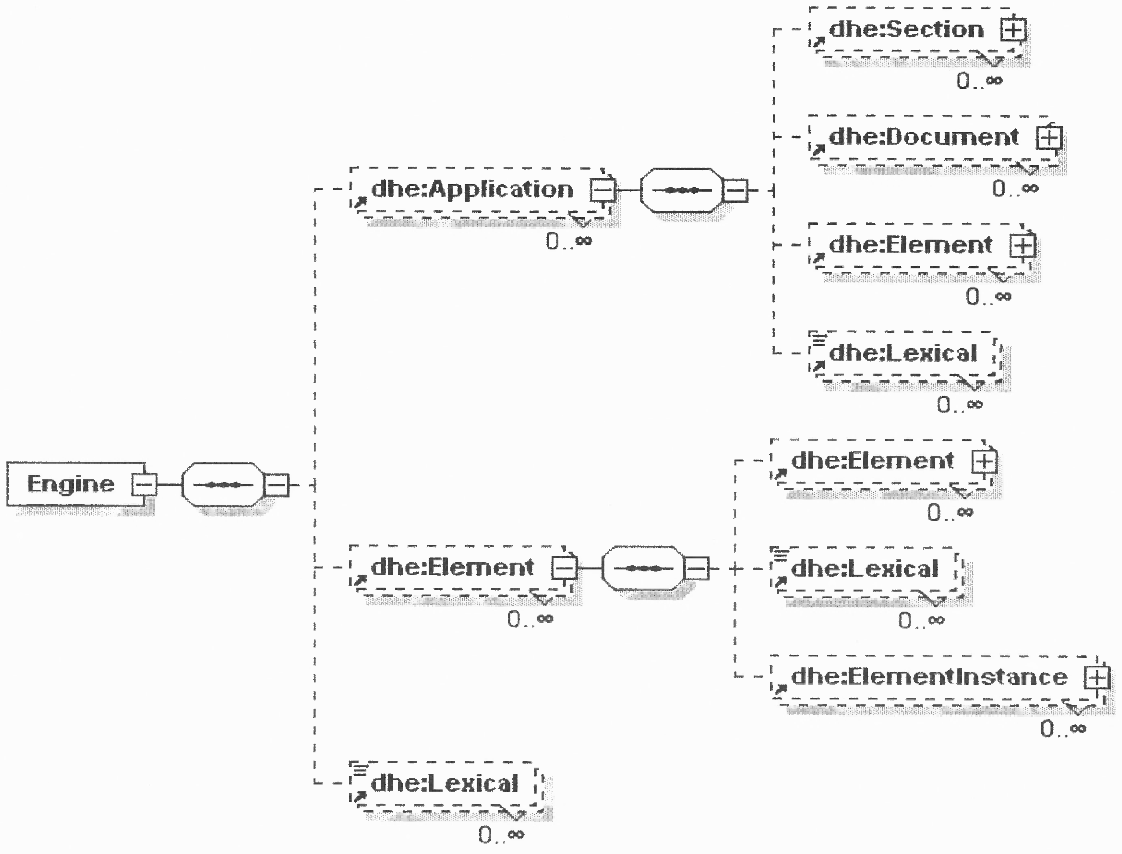


Figure 6.2 DHE-Meta-Information Scope Hierarchy

The XML Schema language if used to define the DHE-MISH :

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://dhe.njit.edu/Meta-InformationScopeHierarchy"
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
xmlns:dhe="http://dhe.njit.edu/Meta-InformationScopeHierarchy">

  <xsd:element name="MISH">
    <xsd:annotation>
      <xsd:documentation>Meta-Information Scope Hierarchy</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Engine" type="xsd:string">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element ref="dhe:Application" minOccurs="0"
             maxOccurs="unbounded"/>
              <xsd:element ref="dhe:Element" minOccurs="0"
             maxOccurs="unbounded"/>
              <xsd:element ref="dhe:Lexical" minOccurs="0"
             maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string" use="fixed"
           value="DHE"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="Application">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="dhe:Section" minOccurs="0"
        maxOccurs="unbounded"/>
        <xsd:element ref="dhe:Document" minOccurs="0"
       maxOccurs="unbounded"/>
        <xsd:element ref="dhe:Element" minOccurs="0" \
        maxOccurs="unbounded"/>
        <xsd:element ref="dhe:Lexical" minOccurs="0"
       maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:string" use="required"/>
    </xsd:complexType>
  </xsd:element>
```

```
<xsd:element name="Section">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="dhe:Section" minOccurs="0"
           maxOccurs="unbounded"/>
            <xsd:element ref="dhe:Document" minOccurs="0"
           maxOccurs="unbounded"/>
            <xsd:element ref="dhe:Element" minOccurs="0"
            maxOccurs="unbounded"/>
            <xsd:element ref="dhe:Lexical" minOccurs="0"
           maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string" use="required"/>
    </xsd:complexType>
</xsd:element>

<xsd:element name="Document">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="dhe:Element" minOccurs="0"
            maxOccurs="unbounded"/>
            <xsd:element ref="dhe:Lexical" minOccurs="0"
            maxOccurs="unbounded"/>
            <xsd:element ref="dhe:DocumentInstance" minOccurs="0"
            maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="type" type="xsd:string" use="required"/>
    </xsd:complexType>
</xsd:element>

<xsd:element name="DocumentInstance">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="dhe:Element" minOccurs="0"
           maxOccurs="unbounded"/>
            <xsd:element ref="dhe:Lexical" minOccurs="0"
            maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="instance" type="xsd:string" use="required"/>
    </xsd:complexType>
</xsd:element>

<xsd:element name="Element">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="dhe:Element" minOccurs="0"
           maxOccurs="unbounded"/>
            <xsd:element ref="dhe:Lexical" minOccurs="0"
            maxOccurs="unbounded"/>
            <xsd:element ref="dhe:ElementInstance" minOccurs="0"
            maxOccurs="unbounded"/>
          </xsd:sequence>
          <xsd:attribute name="type" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="ElementInstance">
        <xsd:complexType>
          <xsd:sequence/>
          <xsd:attribute name="value" type="xsd:string"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="Lexical" type="xsd:string"/>

</xsd:schema>
```

Mapping Rules

Mapping Rules for which their only ancestor is DHE are considered ‘global’ and should be included for any occurrences in any application.

Mapping rules should be semantically typed.

Figure 6.3 shows the structure of the Meta-Information Mapping Rules.

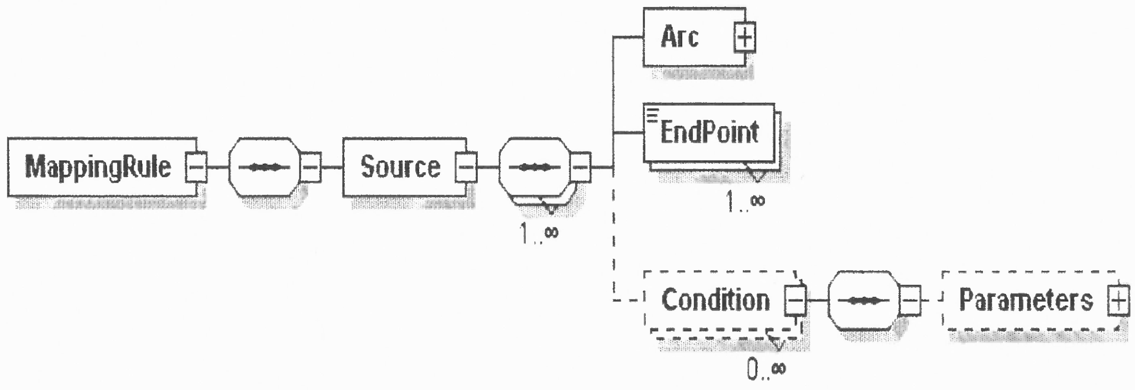


Figure 6.3 DHE-Meta-Information Mapping Rules Structure

XML Schema used to define the DHE-Meta-Information Mapping Rules:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:dhe="http://dhe.njit.edu/Meta-InformationMappingRules"
targetNamespace="http://dhe.njit.edu/Meta-InformationMappingRules"
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">

  <xsd:element name="MIMR">
    <xsd:annotation>
      <xsd:documentation>Meta-Information Mapping Rules</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="MappingRule" minOccurs="0"
          maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Source" minOccurs="1" maxOccurs="1">
                <xsd:complexType>
                  <xsd:sequence minOccurs="1" maxOccurs="unbounded">
                    <xsd:element name="Arc" type="dhe:ArcType"/>
                    <xsd:element name="EndPoint" type="dhe:LocatorType"
                    minOccurs="1" maxOccurs="unbounded"/>
                    <xsd:element name="Condition" minOccurs="0"
                      maxOccurs="unbounded">
                      <xsd:complexType>
                        <xsd:sequence>
                          <xsd:element name="Parameters"
                            type="dhe:ParameterGroup" minOccurs="0"
                            maxOccurs="1"/>
                        </xsd:sequence>
                        <xsd:attribute name="location"
                          type="dhe:LocatorType" use="required"/>
                      </xsd:complexType>
                    </xsd:element>
                  </xsd:sequence>
                  <xsd:attribute name="type" type="dhe:LocatorType"/>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="MRid" type="xsd:string" use="required"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="ArcType">
    <xsd:sequence>
```

```
        <xsd:element name="SemanticType" type="xsd:string" minOccurs="1"
         maxOccurs="unbounded"/>
        <xsd:choice>
            <xsd:element name="CommandSet" minOccurs="1" maxOccurs="1">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="Command" type="xsd:string"
                      minOccurs="1" maxOccurs="1"/>
                        <xsd:element name="Parameters" type="dhe:ParameterGroup"
                            minOccurs="0" maxOccurs="1"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="URI" type="xsd:uriReference"/>
        </xsd:choice>
    </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="LocatorType">
    <xsd:restriction base="xsd:string">
        <xsd:annotation>
            <xsd:documentation>
                This string should comply with the XPath and XPointer syntax as
            defined in their specifications
            </xsd:documentation>
        </xsd:annotation>
    </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="ParameterGroup">
    <xsd:sequence minOccurs="1" maxOccurs="unbounded">
        <xsd:element name="ParamName" type="xsd:string"/>
        <xsd:element name="ParamValue" type="xsd:string" minOccurs="0"
         maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>

</xsd:schema>
```

Log Module (LM)

The Log Module handles the logging of the activity of the Engine. LM will provide every module with the possibility of logging their inputs and/or outputs, based on the values of preferences.

Preferences Module (PM)

The Preferences Module enables users and modules to store and retrieve preferences that can influence the execution and display of results.

There are some preferences that all DHE modules should implement, and there should be a DHE level preference to turn on/off this preferences for every module:

- Debug Preference, modules should check the value of the debug preference when executing, and output debug info when it is turn on. When Debug is 'on', Log Input and Log Output should behave as 'on' too.
- Log Input, if this preference is set to 'on', the corresponding module's log object should record all inputs.
- Log Output, if this preference is set to 'on', the corresponding module's log object should record all outputs.

Exception Handler (EH)

The Exception Handler will provide a common framework for handling and reporting problems during the running of the Engine.

Virtual Documents Module (VDM)

The Virtual Documents Module provides the methods required for the creation, use and storage of virtual documents. The VDM in built on top of the W3C's Document Object Model [DOM 00].

The VDM is an application programming interface (API) for DHE-VirtualDocuments. It defines the logical structure of documents and the way a document is accessed and manipulated. The term "document" is used in the broad sense - increasingly, XML is being used as a way of representing many different kinds of information that may be stored in diverse systems, and much of this would traditionally be seen as data rather than as documents. Nevertheless, XML presents this data as documents, and the DOM may be used to manage this data.

With the VDM, programmers can build DHE-VirtualDocuments, navigate their structure, and add, modify, or delete elements and content. Anything found in a DHE-VirtualDocument can be accessed, changed, deleted, or added using the VDM (with a few exceptions).

One important objective for the Virtual Document Module is to provide a standard programming interface that can be used in a wide variety of environments and applications.

DHE-VirtualDocument

DHE model of execution uses a DHE-VirtualDocument object to store the information required for, and generated by, the processing of a user request.

The DHE-VirtualDocument is made up by the following components shown in Figure 6.4:

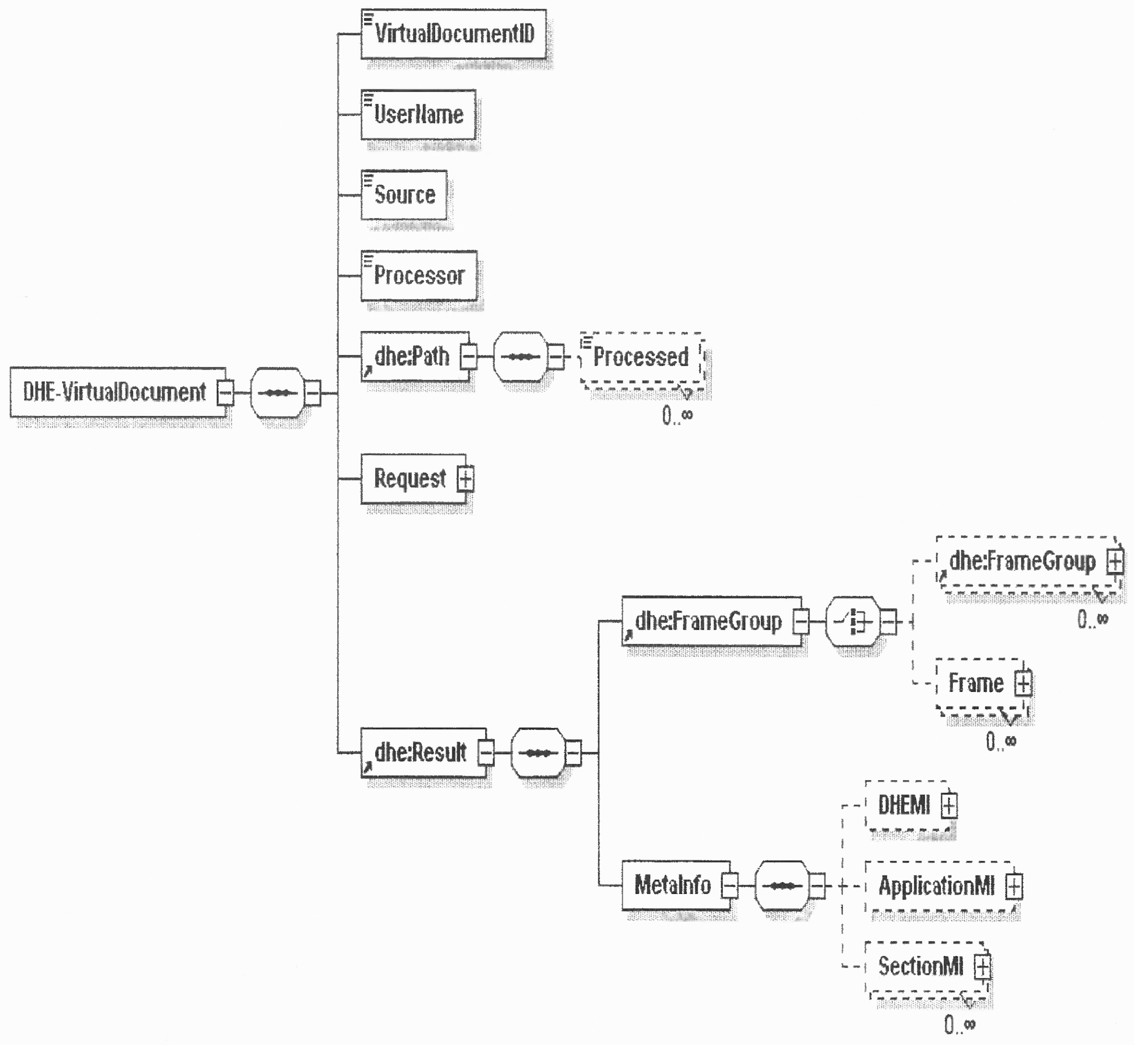


Figure 6.4 DHE-VirtualDocument Structure

The XML Schema language if used to define the DHE-VirtualDocument :

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://dhe.njit.edu/VirtualDocument"
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
xmlns:dhe="http://dhe.njit.edu/VirtualDocument">

  <xsd:element name="DHE-VirtualDocument">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="VirtualDocumentID" type="xsd:string"/>
        <xsd:element name="UserName" type="xsd:string"/>
        <xsd:element name="Source" type="xsd:string"/>
        <xsd:element name="Processor" type="xsd:string"/>
        <xsd:element ref="dhe:Path"/>
        <xsd:element name="Request" type="dhe:MappingRule"/>
```

```
            <xsd:element ref="dhe:Result"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="Path">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Processed" type="xsd:string" minOccurs="0"
            maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="ParameterGroup">
    <xsd:sequence>
        <xsd:element name="ParamName" type="xsd:string"/>
        <xsd:element name="ParamValue" type="xsd:string" minOccurs="0"
         maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:element name="Result">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="dhe:FrameGroup"/>
            <xsd:element name="MetaInfo">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="DHEMI" type="dhe:MIType"
                       minOccurs="0" maxOccurs="1"/>
                        <xsd:element name="ApplicationMI" type="dhe:MIType"
                       minOccurs="0" maxOccurs="1"/>
                        <xsd:element name="SectionMI" type="dhe:MIType"
                       minOccurs="0" maxOccurs="unbounded"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="FrameGroup">
    <xsd:complexType>
        <xsd:choice>
            <xsd:element ref="dhe:FrameGroup" minOccurs="0"
            maxOccurs="unbounded"/>
            <xsd:element name="Frame" type="dhe:FrameType" minOccurs="0"
            maxOccurs="unbounded"/>
        </xsd:choice>
        <xsd:attribute name="Name" type="xsd:string" use="required"/>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="FrameType">
    <xsd:sequence>
        <xsd:element name="Document">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:any namespace="http://www.w3.org/1999/xhtml
                        http://dhe.njit.edu/" processContents="skip"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="MetaInfo">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="DocumentMI" type="dhe:MIType"
                   minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="DocumentInstanceMI" type="dhe:MIType"
                   minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="ElementMI" type="dhe:MIType"
                   minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="ElementInstanceMI" type="dhe:MIType"
                    minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="LexicalMI" type="dhe:MIType"
                   minOccurs="0" maxOccurs="1"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="Name" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="MIType">
    <xsd:sequence maxOccurs="unbounded">
        <xsd:element name="Locator" type="dhe:LocatorType"/>
        <xsd:element name="Relationships" type="dhe:MappingRule"
        minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="MappingRule">
    <xsd:sequence>
        <xsd:element name="Source">
            <xsd:complexType>
                <xsd:sequence maxOccurs="unbounded">
```

```
                    <xsd:element name="Arc" type="dhe:ArcType"/>
                    <xsd:element name="EndPoint" type="dhe:LocatorType"
                    maxOccurs="unbounded"/>
                    <xsd:element name="Condition" minOccurs="0"
                    maxOccurs="unbounded">
                        <xsd:complexType>
                            <xsd:sequence>
                                <xsd:element name="Parameters"
                                    type="dhe:ParameterGroup" minOccurs="0"/>
                            </xsd:sequence>
                            <xsd:attribute name="location" type="dhe:LocatorType"
                          use="required"/>
                        </xsd:complexType>
                    </xsd:element>
                </xsd:sequence>
                <xsd:attribute name="type" type="dhe:LocatorType"/>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="MRid" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="ArcType">
    <xsd:sequence>
        <xsd:element name="SemanticType" type="xsd:string"
         maxOccurs="unbounded"/>
        <xsd:choice>
            <xsd:element name="CommandSet">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="Command" type="xsd:string"/>
                        <xsd:element name="Parameters" type="dhe:ParameterGroup"
                            minOccurs="0"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="URI" type="xsd:uriReference"/>
        </xsd:choice>
    </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="LocatorType">
    <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

</xsd:schema>
```

6.1.2 Application Wrappers (AW)

To integrate an Application with DHE is necessary first to perform a thorough analysis of the structure, inputs, outputs, and relationships inside of it. A methodology like the Relation-Navigation Analysis [Yoo 00] can provide valuable direction for this task.

After the application analysis is completed, the Application Meta-Info Hierarchy (see Meta-Information Module description in 6.1.1 for details) is stored with their corresponding mapping rules and metadata.

The functionality of the AWs has been further decomposed (as from DHE 1.0) to allow a higher degree of standardization. The goal being to make the development of new AWs more straightforward and to foment the re-use of components between them.

A generic DHE NG AW should implement the following functions:

- Application Communications. This module handles the transmission and reception of data between the application and the AW. Application Communications needs to perform two main functions:
    - AWtoApplication. Enable the AW to send a request to the corresponding Application. This module will communicate with the Application through its API.
    - ApplicationtoAW. Should be able to receive the result of a request from the Application and make the data available to the AW Manager.

- DHE Communications. This module should handle the transmission and reception of data between the DHE and the AW. The implementation should allow multiple threads of execution to be run in parallel, each one processing a different DHE-VirtualDoc. DHE Communications needs to perform two main functions:
  - AWtoDHE. Return the DHE-VirtualDoc, with the result of the request, to the DHE Controller.
  - DHEtoAW. Receive the DHE-VirtualDoc with the request from the DHE Controller module and to make it available to other AW processes.
- AW Manager. The main function of the AW Manager is to identify the type of output generated by the Application (i.e. document type), detect the objects and/or elements it has previously identified as having meta-Information, and update the DHE-VD with the results (see description of the DHE-VD in the Virtual Document Section). The AW needs to perform several important tasks:
  - API Transformation. Will take the request embedded in the DHE-VD and translate into the Applications' API. The instruction will be then send to the Application by the AWtoApplication method.
  - Application Document Processing. The processing of the outcome from the Application is a complex task involving:
    - Receive the result (using ApplicationtoAW) and match it with the DHE-VD that contains the request. AWs should be able to handle multiple concurrent user requests. The AW Manager must ensure

that the result of a request made by any particular user session is returned back to the appropriate DHE-VirtualDoc.

- Identify the type of the Application output Document and insert it in the corresponding DHE-VD element.
- Parse the A-Doc to detect the elements of interest and mark them. The proposed method to implement this task uses the W3C's XML Schema specification [XSchema 01] to code the different document types that form the applications outcome set.
- Transform the A-Doc to the DHE Output Document Format [XHTML 00], if necessary, and include it in the DHE-VD. One important difference between the AWs in DHE NG and DMISWs in DHE 1.0 is that AWs should encode the result produced by the application using XML tags. DHE 1.0 uses HTML inside an XML framework. as well as to keep the original, unmodified, output.
- Create and attach the list of element types found in the A-Doc to the DHE-VD.
- Make the DHE-VD with the result available to the DHE Controller.

6.1.3 User Interface Wrappers (UIW)

The UIW constructs the DHE-VirtualDoc without having to go back to MIM. The employment of User Interface Wrappers enables DHE to work with applications in several diverse platforms without having to modify the applications themselves.

Only sub-modules of the generic UIW need to be updated to conform with the particular User Interface.

The functionality of the UIWs has been further decomposed (as from DHE 1.0) to allow a higher degree of standardization. The goal being to make the development of new UIWs more straightforward and to foment the re-use of components between them.

A generic DHE UIW should implement the following functions:

User Interface Communications. This sub-module handles the transmission and reception of data between the UI and the UIW. UI Communications needs to perform two main functions:

UItoUIW. Should be able to receive a request from the User Interface and make the data available to the Session Manager.

UIWtoUI. Will enable the UIW to send the result of a request to the corresponding UI.

DHE-Virtual Document Instantiation. Will create a DHE-VirtualDocument, (using the Virtual Documents Module), and enter the information from the request into it. See the definition of the DHE-VD in the Virtual Document Module section.

DHE Communications. This module should handle the transmission and reception of data between the DHE and the UIW. The implementation should allow multiple threads of execution to be run in parallel, each one processing a different DHE-VirtualDoc. DHE Communications needs to perform two main functions:

UIWtoDHE. Invoke the DHE Controller, and to make the DHE-VirtualDoc (created by UItoUIW) available to it.

DHEtoUIW. Receive the DHE-VirtualDoc back from the Controller module and to make it available to other UIW processes.

UIW Manager. UIWs should be able to handle multiple concurrent user sessions. A single user may have more than one parallel sessions at any point. There may be several users working at the same time too. The UIW Manager must ensure that the result of a request made by any particular user session is returned back to the appropriate UI instance.

The UIW Manager should keep the DHE-VD in store until there is another request from that particular session (or until the session is terminated). When a new request arrives, the UIW Manager must determine if the operation requested can be fulfilled using the meta-Information included in the current DHE-VD for that session

UI-Document Formatter. The Formatter's task is to create the output UI document based on the contents of the DHE-VD. The Formatter will make sure that the UI-Document is coded in the appropriate format for the UI. The UI-Document Formatter also needs to make available DHE's additional functionally for user's utilization. It is DHE's responsibility to include all the information necessary to perform the extra functions in the DHE-VC.

Security. The UIW must implement a login procedure to prevent unauthorized access to DHE.

6.2 Summary

DHE NG's architecture aims to correct some of the problems found in the first version of DHE as well as include the infrastructure required to implement new features.

The main goals of the new design are:

- Improve performance: making DHE more tightly coupled, storing information locally, maximizing the metainformation included in the results.
- Additional Hypermedia functionalities: guided tours, manual linking, annotations.
- Support Features: regeneration and saving, object types and identifiers, target areas, conditions, error recovery and debugging, administrative tools.

The DHE NG architecture is similar to DHE 1.0, with the Engine as middleware between the applications and the user interface. Wrappers habilitate the interaction with applications and user interfaces.

The main components of DHE NG are: user interfaces, user interfaces wrappers, controller, applications wrappers, and applications. The Controller includes a process manager and a meta-information module.

The Meta-Information Scope Hierarchy allows users to define relationships between elements at several different levels: engine, application, section(s), document, element, lexical, and instance. The Meta-Information Scope Hierarchy has a set of rules and constraints that should be enforced.

The Virtual Documents Module provides the methods required for the creation, use and storage of virtual documents. It defines the logical structure of documents and the way a document is accessed and manipulated. One important objective for the Virtual Document Module is to provide a standard programming interface that can be used in a wide variety of environments and applications.

CHAPTER 7

RESULTS AND FUTURE RESEARCH

7.1 Contributions

The Dynamic Hypermedia Engine is a modular distributed middleware able to enhance Information Systems with metadata, along with hypermedia structuring, navigation and annotative functionality.

DHE is the only approach we know that provides automated linking and metadata services in a generic manner, based on the application semantics (as opposed to search or lexical analysis), without altering applications. It is uniquely suited to support analytical and technical applications that generate the contents of their displays dynamically.

The DHE prototype offers a solution to the problem of how to include linking and metadata functionality to a virtual document. The updated DHE architecture will become the new test bed for the implementation of new hypermedia functionalities, and offers a solution for several of the problems related with the management of dynamically generated documents.

The contributions are listed as follows:

- Conceptual Framework. The conceptual framework states the problem and the proposed solution. In particular it defines the central idea of this research:
    - The automatic creation of links and metadata services can be done in a generic way, based on the application semantics and structure.

- DHE 1.0. DHE 1.0 is a 'proof-of-concept' prototype that shows the possible benefits of several hypermedia functionalities. DHE also serves as a test bed for additional hypermedia research. Such research might include empirical experiments and field studies.
- DHE NG: The next generation of the Dynamic Hypermedia Engine extends the DHE 1.0 architecture to be able to support non-standard hypermedia functionalities and regeneration.

The DHE 1.0 Implementation has the following features:

- Provides a [Web] interface
- Automated meta-Information and hypermedia functionality
- Support for Virtual Documents
- Mapping rules represent an application internal structure
- Distributed modular architecture
- Wrappers to integrate applications with minimal changes
- XML, RDF, XLink, HTML
- Java 1.2, RMI, Servlets

The DHE NG extends DHE 1.0 with the following new features:

- Hypermedia and Virtual Documents
- Meta-Information Scope Hierarchy
- Document and Element Identifiers
- Regeneration
- Locators
- Bi-directional Linking

- Saving
- Additional Hypermedia Functionality
- Guided Tours
- Annotations
- Manual Linking

7.2 Future Research

This research touches many diverse areas and can, thus, be extended in several different directions:

- Use of Schemas as a source of semantic information
- Filtering and ranking of Meta-Information
- User Interface and User Processes within DHE
- Virtual Documents Management
- DHE as an Application Development Environment
- Finding relationships dynamically through data mining
- Integrating different applications
- Qualitative and Quantitative Experiments
- Semantic Web

7.2.1 Use of schemas as a source of semantic information.

XML has provided the Web with a powerful data format to express complex data structures. However, most existing documents are not in XML format. To be able to create semantic applications (See the Semantic Web section later in this Chapter) it's necessary to add semantic information to those documents that currently do not have it, including those which are dynamically generated.

Conversion between non-XML file data and XML can be done via generic conversion tools or custom scripts. Generic conversion tools are schema-driven. Such conversion tool uses the schema of the file format in order to parse the file and convert it to an XML document.

Currently, the definition of a schema for non-XML files requires a lot of manual work: analysis of the file structure, analysis of the data type and element attributes of data in the file, and a mapping of that structure into semantically meaningful markup.

The scale of an up-conversion project could be very large and requires a lot of difficult manual conversion efforts, because many documents do not have hierarchical structure. Finding and adding semantic information automatically to documents or files is very difficult and has not been satisfactorily addressed before.

One of the most fundamental features of the DHE project is its ability to add semantic information to dynamically generated, 'virtual', documents. DHE needs to parse and identify the various element with meta-information which are present in a particular instance of a document.

To be able to add semantic information to virtual documents, it is necessary to analyze their structures. The result of such analysis would be a set of 'document types' as well as a document schema for each one of those different types. The document schema could be expressed by an XML language like XML Schema. The schema should contain enough information to enable an application wrapper to parse the different documents and generate the semantic markup. In such case, most of the meta-information could be specified at the schema level.

7.2.2 The Semantic Web

"The World Wide Web is a universal information space. As a medium for human exchange, it is becoming mature, but we are just beginning to build a space where automated agents can contribute -- just beginning to build the Semantic Web."

Tim Berners-Lee, W3C Director [Berner-Lee 99]

The architecture of the World Wide Web provides users with a simple hypertext interface to a variety of remote resources, from static documents purely for human consumption to interactive data services. Now that the Web has reached critical mass as a medium for human communication, the next phase is to build the "Semantic Web". The Semantic Web is a Web that includes documents, or portions of documents, describing explicit relationships between things and containing semantic information intended for automated processing by our machines.

The concept of machine-understandable documents does not imply some kind of magical artificial intelligence system that will allow machines to comprehend human language. It only indicates a machine's ability to solve a well-defined problem by performing well-defined operations on existing well-defined data. Instead of asking machines to understand people's language, it involves asking people to make the extra effort.

PICS was designed as a first step toward generalized labels that would allow any party in the Web to make claims about the qualities of resources: endorsements, terms and conditions for use, and so on. The W3C Metadata Activity addresses the necessary work to complete the picture: structured labels, rules, integration with digital signatures. The PICS label design was generalized to a model of information as directed labeled graphs (DLGs). This was known as the RDF model, and a serialization was defined in XML syntax.

Traditionally, both documents and databases have been strongly typed; that is, the producer and consumer have prior agreement on the structure of the information units. But this by itself is not sufficient for the developing of the Semantic Web. The Semantic Web must permit distributed communities to work independently to increase the Web of understanding, adding new information without insisting that the old be modified. This approach allows the communities to resolve ambiguities and clarify inconsistencies over time while taking maximum advantage of the wealth of backgrounds and abilities reachable through the Web. Therefore the Semantic Web must be based on a facility that can expand as human understanding expands. This facility must be able to capture information that links independent representations of overlapping areas of knowledge.

Incremental decentralized development of Semantic Web applications requires documents to be able to contain an ad hoc mixture of features from multiple application domains. The combinatory issues make it impractical to predefine document types that encompass all the possible vocabulary sets. The XML Namespace facility will allow this vocabulary mix-in.

In the same way that HTML and XML Linking allow authors to lead readers from any place in the Web to any other place in the Web, data in the Semantic Web must be able to relate anything to anything.

To encompass the universe of network-accessible information [Berner-Lee 92], the Semantic Web must provide a way of exposing information from different systems. These systems may use a variety of internal data models so this implies a requirement for some generic concept of data at a low level that is in common between each system.

Another challenge of the Semantic Web, then, is to support the mapping of the existing and future systems onto the Web, preserving the universality of the Web and also the properties of the local systems. Optimizations - such as being able to enumerate and index all objects of a given type - that are important to the local operation of a system do not scale to the Web.

The mechanism adopted in RDF to manage the expression of constraints is to make all objects, all relationships, all types, and even all assertions be "first class objects" on the Web. That is; they have their own URIs and are not constrained in the fundamental level to be combined in any particular way. Giving first class identifiers to types, relationships, and assertions will allow the Semantic Web to make assertions about itself.

All statements found on the Web occur in some context. Trustworthiness is evaluated by, and in the context of, each application that processes the information found on the Web.

Just as the design of the Web sacrificed link integrity for scalability, the "all knowledge about my thing is contained here" notion cannot hold when databases and objects are exported to the Web. A great benefit to relaxing this assumption will be that, just as hypertext links connect different information systems, the Semantic Web will connect data from vastly different systems, allowing complex and far-reaching processing of a wide store of available data.

Other concerns at this point are raised about the relationship to knowledge representation systems that have already tried some of these concepts before, with projects such as [KIF] and cyc [Lenat 95]. They should feed the semantic Web with design experience and the Semantic Web may provide a source of data for reasoning engines developed in similar projects.

Many knowledge representation systems had a problem merging or interrelating two separate knowledge bases, as the model was that any concept had one and only one place in a tree of knowledge. They therefore did not scale, or pass the test of independent invention. The RDF world, by contrast is designed for this in mind, and the retrospective documentation of relationships between originally independent concepts.

Knowledge representation is a field which currently seems to have the reputation of being initially interesting, but which did not seem to shake the world to the extent that some of its proponents hoped. It made sense but was of limited use on a small scale, but never made it to the large scale. That is exactly the state the hypertext field was in before the Web. Each field had made certain centralist assumptions -- if not in the philosophy, then in the implementations, which prevented them from spreading globally. But each field was based on fundamentally sound ideas about the representation of knowledge. The Semantic Web is what the Web will become if the same globalization process that the Web initially did to Hypertext is performed to Knowledge Representation.

The W3C's Semantic Web Activity [W3C SW] has been recently established to serve a leadership role, in both the design of enabling specifications and the open, collaborative development of technologies that support the automation, integration and reuse of data across various applications. To facilitate this goal, the Semantic Web Activity builds upon the existing foundation work accomplished by the W3C Metadata Activity.

The Semantic Web Activity connects with and augments other W3C work efforts. In particular, the Semantic Web work will inform and support further work on:

- harvesting semantic relationships in RDF form from documents using XLink hyperlinking,
- describing the signer's intent when using XML Signatures,
- semantics associated with data usage practices and policies in P3P, and profiles in CC/PP,
- annotation to improve the accessibility of Web pages to persons with disabilities in WAI,
- investigations into the use of the XML DOM for non-tree-structured data,
- document structure transformations with XSLT,

- integrating other generic data models and type hierarchies with XML schema,
- incorporating RDF semantics in an XML schema,
- articulation of characteristics of resources that support the identified resources as defined by the URI Activity.

7.3 Summary

The Dynamic Hypermedia Engine is able to enhance Information Systems with metadata, and hypermedia structuring, navigation and annotative functionality.

DHE provides automated linking and metadata services in a generic manner, based on the application semantics, without altering applications.

The DHE prototype gives a solution to the problem of how to include linking and metadata functionality to a virtual document. The updated DHE architecture will become the new test bed for the implementation of new hypermedia functionalities, and offers a solution for several of the problems related with the management of dynamically generated documents.

The main contributions are: a conceptual framework, DHE 1.0 design and implementation, and the DHE NG architecture.

The DHE 1.0 Implementation has the following features: provides a [Web] interface, automated meta-Information and hypermedia functionality, support for virtual documents, mapping rules representing an application internal structure, distributed modular architecture, wrappers to integrate applications with minimal changes, use of Java and XML standards.

The DHE NG extends DHE 1.0 with the following new features: new hypermedia functionality and virtual documents, the Meta-Information Scope Hierarchy, document and element Identifiers, regeneration, locators, bi-directional linking. The new hypermedia functionalities include: guided tours, annotations, and manual linking.

The possible topics for future research are: use of schemas as a source of semantic information, filtering and ranking of Meta-Information, user Interface and user processes within DHE, virtual document management, DHE as development environment, relationships mining, integration of new applications, qualitative and quantitative experiments, and the semantic Web.

REFERENCES

[Anderson 96]	Anderson, K.; *Providing automatic support for extra-application functionality*, Proceedings of the Second International Workshop on Incorporating Hypertext Functionality Into Software Systems (HTF II), March 1996.
[Anderson 97]	Anderson, K. Integrating Open Hypermedia Systems with the World Wide Web. Hypertext'97 Proceedings, ACM Press, New York, NY, April 1997, 157-166.
[Arocena 98]	Arocena, G.; Mendelzon, A.; *WebOQL: Restructuring Documents, Databases, and Webs* Proceedings of the 14th International Conference on Data Engineering, Orlando, Florida, 24-33, February 1998.
[Ashman 97]	Ashman, H.; Garrido, A.; Oinas-Kukkonen, H.; *Hand-made and Computed Links, Precomputed and Dynamic Links* in Proceedings of Multimedia '97 (HIM '97), Germany, 191-208, 1997.
[Ashman 99]	Ashman, H.; Oinas-Kukkonen, H.; Bieber, M.; *Hypertext Functionality: introduction to the special issue*, Journal of Digital Information, Special issue guest editorial, Volume 1 issue 4, February 1999. http://jodi.ecs.soton.ac.uk/Articles/v01/i04/editorial/
[Bennett 95]	Bennett, K.; *Legacy systems: coping with success*, IEEE Software, January 1995, 19-23.
[Berners-Lee 89]	Berners-Lee, T.; *Information Management: A Proposal*, CERN March 1989, May 1990. http://www.w3.org/History/1989/proposal.html
[Berners-Lee 92]	Berners-Lee, T., et al.; *World-Wide Web: The Information Universe Electronic Networking: Research, Applications and Policy*, Meckler, Vol 1 No 2, Spring 1992.
[Berners-Lee 99]	Berners-Lee, Tim; Connolly, Dan; Swick, Ralph R.; *Web Architecture: Describing and Exchanging Data*, W3C Note, 7 June 1999. http://www.w3.org/1999/04/WebData

[Beynon-Davies 94]	Beynon-Davies, P.; Tudhope, D.; Taylor, C.; Jones, J.; *A Semantic Database approach to Knowledge-Based Hypermedia Systems* in Information and Software Technology 36, 6, 323-329, 1994.
[Bieber 92]	Bieber, M.; And Kimbrough, S.; *On generalizing the concept of hypertext*, Management Information Systems Quarterly 16(1), 1992, 77-93.
[Bieber 94]	Bieber, M. And Kimbrough, S.; *On the logic of generalized hypertext*, Decision Support Systems 11, 1994, 241-257.
[Bieber 95a]	Bieber, M.; Isakowitz, T. (eds.); *Designing Hypermedia Applications*, Special issue of the Communications of the ACM 38(8), 1995.
[Bieber 95b]	Bieber, M.; Kacmar, C.; *Designing hypertext support for computational applications*, Communications of the ACM 38(8), 1995, 99-107.
[Bieber 97a]	Bieber, M.; Vitali, F.; *Toward Support for Hypermedia on the World Wide Web*, IEEE Computer, 30(1), 1997, 62-70.
[Bieber 97b]	Bieber, M.; Vitali, F.; Ashman, H.; Balasubramanian, V.; Oinas-Kukkonen, H.; *Fourth Generation Hypermedia: Some Missing Links for the World Wide Web*. International Journal of Human Computer Studies, Vol. 47, 31-65.
[Bieber 97c]	Bieber, M.; *Supplementing Applications with Hypermedia*, Technical Report, Version 1.4 - 8/12/97. http://www.cis.njit.edu/~bieber/pub/supp/supp.html
[Bieber 00]	Bieber, M.; Oinas-Kukkonen, H.; Balasubramanian, V.; *Hypertext Functionality*, ACM Computing Surveys .(forthcoming)
[Bodner 97]	Bodner, R.; Chignell, M.; Tam, J.; *Website authoring using dynamic hypertext*, Proceedings of Webnet'97, Toronto: Association for the Advancement of Computing in Education, (1997), 59-64, 1997.
[Bullock 98]	Bullock, J.; Goble, C.; *TourisT: The Application of a Description Logic based Semantic Hypermedia System for Tourism* Proceedings of ACM Hypertext '98, Pittsburgh PA, 132-141, June 1998.

[Bush 45]	Bush, Vannevar; *As we may think*, The Atlantic Monthly, July 1945. http://www.isg.sfu.ca/~duchier/misc/vbush/vbush.shtml
[Campbell 88]	Campbell, B.; Goodman, J.; *HAM: a general purpose hypertext abstract machine*, Communications of the ACM 31(7), 1988, 856-861.
[Carr 98]	Carr, L. A.; Hall, W.; Hitchcock, S.; *Link Services or Link Agents?*, Proceedings of ACM Hypertext '98, Pittsburgh PA, 113-122. http://acm.org/pubs/citations/proceedings/hypertext/276627/p113-carr/
[Carr 99]	Carr, L.; Hall, W.; DeRoure, D.; *The Evolution of Hypermedia Link Services*, ACM Computing Surveys, Symposium on Hypertext and Hypermedia, 1999.
[Charoenkitkarn 95]	Charoenkitkarn, N.; Chignell, M.; Golovchinsky, G.; *Interactive Exploration as a Formal Text Retrieval Method: How Well can Interactivity Compensate for Unsophisticated Retrieval Algorithms?* Proceedings of the Third Text Retrieval Conference (TREC-3), Gaithersburg, Maryland, 179-199, 1995.
[Conklin 87]	Conklin, J.; *Hypertext: An Introduction and Survey*, IEEE Computer, 20(9), 17-41, 1987.
[Cunliffe 97]	Cunliffe, D; Taylor, C; Tudhope, D.; *Query-based Navigation in Semantically Indexed Hypermedia*, Proceedings of ACM Hypertext 97, Southampton, UK, 87-95, April 1997.
[Davenport 94]	Davenport, Tom; *Saving IT's Soul: Human-Centered Information Management*, Harvard Business Review, March/April 1994, pp. 119 –131.
[Davis 92]	Davis, H.; Hall, W.; Heath, I.; Hill, G.; Wilkins, R.; Towards an integrated information environment with open hypermedia systems, Proceedings of the ACM Conference on Hypertext (Milan, Nov. 1992) 181-190.
[Davis 94]	Davis, H.; Knight, S.; Hall, W.; *Light hypermedia link services: a study of third party application integration*, Proceedings of the Fifth ACM Conference on Hypermedia Technologies, Edinburgh, Scotland, 1994, 158-166.

[Davis 98] Davis, H.; *Referential Integrity of Links in Open Hypermedia Systems*, Proceedings of ACM Hypertext '98 , Pittsburgh, PA, 207-216, June 1998.

[Davis 99] Davis, H.; Hypertext Link Integrity, ACM Computing Surveys, Symposium on Hypertext and Hypermedia, 1999.

[DC] Dublin Core Metadata Element Set Home Page. http://purl.oclc.org/dc/

[De Bra 98] De Bra, P. *History of hypertext and hypermedia*, in Hypermedia Structures and Systems course at the Eindhoven University of Technology 1998. <http://wwwis.win.tue.nl:8001/2L690/cgi/get/a1/history.html>

[Dobie 99] Dobie, M; Tansley, R.; Joyce, D.; Weal, M.; Lewis, P.; Hall, W.; *A Flexible Architecture for Content and Concept-based Multimedia Information Exploration*, Proceedings of the Second UK Conference on Image Retrieval, BCS Electronic Workshops in Computing, 1999.

[DOM 00] W3C Document Object Model (DOM) Level 2 Core Recommendation. http://www.w3.org/TR/DOM-Level-2-Core/

[DSig] Digital Signature Initiative. http://www.w3.org/DSig/

[Engelbart 63] Engelbart, Douglas C.; *A Conceptual Framework for the Augmentation of Man's Intellect*, Vistas in Information Handling, Howerton and Weeks [Editors], Spartan Books, Washington, D. C., 1963, pp. 1-29.

[Fernandez 98] Fernandez, M.; Florescu, D.; Kang, J.; Levy, A.; Suciu, D.; *Catching the Boat with Strudel: Experiences with a Web-Site Management System* Proceedings of ACM SIGMOD '98, Seattle, WA, 414-425, June 1998.

[Finkelstein 99] Finkelstein, C.; Aiken, P.; *Building Corporate Portals with XML*, McGraw-Hill, 1999.

[Florescu 98] Florescu, D.; Levy, A.; Mendelzon, A.; *Database techniques for the world-wide web: A survey* in SIGMOD Record 27, 3, 59-74, 1998.

[Garrido 96]	Garrido, A.; Rossi, G.; *A framework for extending object-oriented applications with hypermedia functionality*, The New Review of Hypermedia and Multimedia 2, 1996, 25-41 ftp://www-lifia.info.unlp.edu.ar/pub/papers/garrido/hypermed.ps.gz
[Garzotto 91]	Garzotto, F.; Paolini, P.; *HDM - a Model for the Design of Hypertext Applications*, Proceedings of the Third ACM Conference on Hypertext, 1991, pp. 313-328.
[Garzotto 95]	Garzotto, F.; Mainetti, L.; Paolini, P.; *Hypertext Design, Analysis, and Evaluation Issues*, Communications of the ACM, Aug. 1995, pp. 74-86.
[Golovchinsky 93]	Golovchinsky, G.; Chignell, M.; Queries-r-links: Graphical Markup for Text Navigation, Proceedings of INTERCHI '93, Amsterdam: The Netherlands, 454-460, 1993.
[Grønbæk 96]	Grønbæk, K; Trigg, R.; *Toward a Dexter-Based Model for Open Hypermedia: Unifying Embedded References and Link Objects*, Proceedings of ACM Hypertext 96, Washington DC, 149-160.
[Grønbæk 00]	Grønbæk, K.; Sloth, L.; Bouvin, N.; Open Hypermedia as User Controlled Meta Data for the Web, Proceedings of the Ninth International World Wide Web Conference, Amsterdam 2000.
[Halasz 94]	Halasz, F.; Schwartz, M.; The Dexter hypertext reference model, Communications of the ACM 37(2), 1994, 30-39.
[Hall 96]	Hall, W.; Davis, H.; Hutchings, G.; *Rethinking Hypermedia, The Microcosm Approach*, Kluwer Academic, Dordrecht, The Netherlands, 1996.
[HICSS 93]	Bieber, M.; Isakowitz, T. (eds); *Hypermedia in Information Systems and Organizations*, Proceedings of the 26th Hawaii International Conference on System Sciences, volume 3 (IEEE Press: Washington, D.C.) 1993.
[HICSS 98]	Isakowitz, T.; Bieber, M. (eds); *Web Information Systems*, Proceedings of the 31st Hawaii International Conference on System Sciences, volume 4 (IEEE Press: Washington, D.C.) 1998.

[Hirata 96] Hirata, K.; Hara, Y.; Takano, H.; Kawasaki, S.; *Content-Oriented Integration in Hypermedia Systems*, Proceedings of ACM Hypertext '96, Washington, DC, 11-21, March 1996.

[HTF 94] Bieber, M.; Oinas-Kukkonen, H. (eds); *Proceedings of the First International Workshop on Incorporating Hypertext Functionality Into Software Systems (HTF I)*, at the European Conference on Hypermedia Technologies (ECHT94), Edinburgh, September 1994.

[HTF 96] Ashman, H.; Balasubramanian, V.; Bieber, M.; Oinas-Kukkonen, H. (eds); *Proceedings of the Second International Workshop on Incorporating Hypertext Functionality Into Software Systems (HTF II)*, at the Hypertext96 Conference, Bethesda, March 1996.

[HTF 97] Ashman, H.; Balasubramanian, V.; Bieber, M.; Oinas-Kukkonen, H. (eds); *Proceedings of the Third International Workshop on Incorporating Hypertext Functionality Into Software Systems (HTF III)*, at the Hypertext97 Conference, Southampton, UK, April 1997.

[HTF 98a] Watters, C.; Vitali, F. (eds); *Proceedings of the Fourth International Workshop on Incorporating Hypertext Functionality Into Software Systems (HTF IV)*, at the 7th International World Wide Web Conference, Brisbane, April 1998.

[HTF 98b] Rossi, G; Ziv, H. (eds); *Proceedings of the Fifth International Workshop on Incorporating Hypertext Functionality Into Software Systems (HTF V)*, at the International Conference on Software Engineering, Kyoto, April 1998.

[HTF 98c] Kuutti, K; Oinas-Kukkonen H. (eds); *Proceedings of the Seventh International Workshop on Hypertext Functionality: Organizational Memory Systems and HTF (HTF VII)*, at the International Conference on Information Systems (ICIS '98), Helsinki, December 1998.

[HTML 98] Hypertext Markup Language 4.0 Specification. http://www.w3.org/TR/REC-html40/

[HyperCard] HyperCard http://www.apple.com/hypercard/

[HyTime]	Hypermedia/Time-based Structuring Language (HyTime). http://www.ornl.gov/sgml/wg8/docs/n1920/html/n1920.html
[Ianella 97]	Iannella, R.; Waugh, A.; *Metadata: Enabling the Internet*, CAUSE97 Conference, Melbourne, Australia, 13-16 April 1997.
[Isakowitz 95]	Isakowitz, T.; Schocken, S.; Lucas, Jr, HC.; *Toward a logical / Physical theory of Spreadsheet Modeling*, ACM Transactions on Information Systems, Journal. Vol. 13, No. 1, January 1995 pp 1-37.
[JOC 96]	Bieber, M.; Isakowitz, T. (eds); *Hypermedia in Information Systems and Organizations*. Journal of Organizational Computing and Electronic Commerce, Vol. 6, No. 3, 1996.
[JoDI 99]	Ashman, H.; Oinas-Kukkonen, H.; Bieber, M. (eds.); *Special issue on Hypertext functionality*; Journal of Digital Information, Volume 1, Issue 4, February 1999. http://jodi.ecs.soton.ac.uk/View/Index/v01/i04/
[Kacmar 93]	Kacmar, C.; Supporting hypermedia services in the user interface, Hypermedia 5(2), 1993, 85-101.
[Kacmar 95]	Kacmar, C.; *A process approach for providing hypermedia services to existing, non-hypermedia applications*, Electronic Publishing: Organization, Dissemination, and Distribution 8(1), March 1995, 31-48.
[KIF]	Knowledge Interchange Format http://www.csee.umbc.edu/kse/kif/
[Kimbrough 79]	Kimbrough, S.; On the reduction of genetics to molecular biology, Philosophy of Science 46(3), 1979, 389-406.
[Lagose 96]	Lagoze, Carl; Lynch, Clifford; Daniel, Ron, Jr.; *The Warwick Framework: A Container Architecture for Aggregating Sets of Metadata*, Cornell Computer Science Technical Report TR96-1593. June, 1996. http://cs-tr.cs.cornell.edu/Dienst/UI/2.0/Describe/ncstrl.cornell/TR96-1593
[Legget 94]	Leggett, J.; Schnase, J.; *Viewing Dexter with open eyes*, Communications of the ACM 37(2), 1994, 77-86.

[Lenat 95]	Lenat, D. B.; *Cyc: A Large-Scale Investment in Knowledge Infrastructure*, Communications of the ACM 38, no. 11, November 1995. See also: http://www.cyc.com/
[Lewis 99]	Lewis, P.; Hall, W.; Carr, L.; DeRoure, D.; The Significance of Linking, ACM Computing Surveys, Symposium on Hypertext and Hypermedia, 1999.
[Marshall 94]	Marshall, C.; Shipman, F.; Coombs, J.; *VIKI: spatial hypertext supporting emergent structure*, European Conference on Hypermedia Technologies'94 Proceedings (Edinburgh, Scotland; September 1994), ACM Press, 13-23.
[Marshall 97]	Marshall, C.; Shipman, F.; *Spatial Hypertext and the Practice of Triage* Proceedings of ACM Hypertext `97, Southampton, UK, 124-133, April 1997.
[MDC]	MetaData Coalition. http://www.mdcinfo.com/
[MDIS 97]	Meta Data Interchange Specification. http://www.mdcinfo.com/MDIS/MDIS11.html
[MOF 97]	Meta Object Facility Specification. http://www.omg.org/cgi-bin/doc?ad/97-08-14.pdf
[Multicosm]	Dynamic Link Service (DLS) Technology Multicosm Ltd. In Chilworth Science Park, Southampton, 1999. http://www.multicosm.com/Technology/TechDynLink.htm
[Nanard 91]	Nanard, J.; Nanard, M.; *Using structured types to incorporate knowledge in hypertext*, Proceedings of ACM Hypertext '91, San Antonio, TX, 329-344, December 1991.
[Nanard 95]	Nanard, J.; Nanard, M. *Hypertext design environments and the hypertext design process*, Communications of the ACM 38(8), 1995, 49-56.
[Nelson 65]	Nelson, Theodore H.; *A File Structure for the Complex, the Changing and the Indeterminate*, proceedings of the ACM 20th national conference 1965.

[Nelson 74]	Nelson, Theodore H.; *Computer Lib/Dream Machines*, Mindful Press 1974.
[Nelson 81]	Nelson, Theodore H.; *Literary Machines*, self-published 1981.
[Newell 73]	Newell, A.; Simon, H. A.; *Human Problem Solving*, Englewood Cliffs, NJ: Prentice-Hall, 1973.
[Nielsen 90]	Nielsen, J.; *The Art of Navigating through Hypertext*, Communications of the ACM, 33(3), 287-310, 1990.
[Nürnberg 98]	Nürnberg, P.; Leggett, J.; A Vision for Open Hypermedia Systems, Journal of Digital Information, Volume 1, issue 2, January 1998 . http://jodi.ecs.soton.ac.uk/Articles/v01/i02/Nurnberg/
[Nürnberg 97]	Nürnberg, P.; Leggett, J.; Schneider, E.; *As We Should Have Thought* Proceedings of ACM Hypertext '97, Southampton, UK, 96-101, April 1997.
[OHSWG]	Open Hypermedia Systems Working Group. http://www.ohswg.org/
[OIM 98]	Open Information Model. http://www.mdcinfo.com/OIM/index.html
[Oinas-Kukkonen 95]	Oinas-Kukkonen, H.; *Developing Hypermedia Systems - the Functionality Approach*, Proceedings of the Second Basque International Workshop on Information Technology (BIWIT í95), keynote address, IEEE Computer Society Press, 2-8.
[OMG]	Object Management Group. http://www.omg.org
[PICS]	Platform for Internet Content Selection. http://www.w3.org/PICS/
[Puttress 90]	Puttress, J.; Guimaraes, N.; *The toolkit approach to hypermedia. Hypertext: Concepts, Systems and Applications*, Proceedings of ECHT'90, Versailles, November 1990, Cambridge University Press, 25-37.
[P3P]	Platform for Privacy Preferences P3P Project. http://www.w3.org/P3P/

[QuickClick]	QuickClick. http://www.quickclick.com/
[RDF 99]	Resource Description Framework Specification. http://www.w3.org/TR/PR-rdf-schema/
[Rivlin 94]	Rivlin, E.; Botafogo, R.; Shneiderman, B.; *Navigating in Hyperspace: Designing a Structure-Based Toolbox*, Communications of the ACM, Feb. 1994, pp. 87-108.
[Rizk 92]	Rizk, A. And Sauter, L.; *Multicard: an open hypermedia system*, Proceedings of the ACM Conference on Hypertext (Milan, Nov. 1992) 4-10.
[Sherman 90]	Sherman, M.; Hansen, W.; Mcinerny, M.; And Neuendorfer, T.; *Building hypertext on a multimedia toolkit: an overview of the Andrew toolkit hypermedia facilities*, Proceedings of European Conference on Hypertext (ECHT) '90 (Cambridge University Press, Versailles, Nov. 1990) 13-24.
[Shipman 99]	Shipman, F.; Marshall, C; *Spatial Hypertext: An Alternative to Navigational and Semantic Links*, ACM Computing Surveys, Symposium on Hypertext and Hypermedia, 1999.
[SIGWEB]	ACM Special Interest Group on Hypertext, Hypermedia and the Web. http://www.acm.org/sigweb/
[SGML 86]	ISO 8879. Information Processing -- Text and Office Systems - Standard Generalized Markup Language (SGML), 1986. http://www.iso.ch/cate/d16387.html
[Tam 97]	Tam, J.; Bodner, R.; Chignell, M.; *Dynamic hypertext benefits novices in question answering*, Proceedings of the Human Factors and Ergonomics Society 41st Annual Meeting, (1997), 350-354, 1997.
[Tudhope 99]	Tudhope, D.; Cunliffe, D.; *Semantically-Indexed Hypermedia: Linking Information Disciplines* ACM Computing Surveys, Symposium on Hypertext and Hypermedia, 1999.
[UML 97]	UML Specification. http://www.omg.org/cgi-bin/doc?ad/97-08-03.pdf

[Vitali 00]	Vitali, F.; Bieber, M.; *Hypermedia on the Web: What Will It Take?*, ACM Computing Surveys (forthcoming). http://www.cis.njit.edu/~bieber/pub/acmcs/cs-vb.html
[Watters 99]	Watters, C.; Shepherd, M.; *Research Issues for Virtual Documents, Workshop on Virtual Documents*, Hypertext Functionality and the Web at the Eighth International World Wide Web Conference, May 1999 - Toronto, Canada. http://www.cs.unibo.it/~fabio/VD99/shepherd/shepherd.html
[Waterworth 91]	Waterworth, J.; Chignell, M.; *A model of information exploration* in Hypermedia 3(1), (1991), 35-58, 1991.
[Weibel 95]	Weible, S.; Godby, J.; Miller, E.; Daniel, R.; *OCLC/NCSA Metadata Workshop Report*, http://www.oclc.org:5046/oclc/research/conferences/metadata/dublin_core_report.html
[Whitehead 97]	Whitehead, E. J.; *An architectural model for application integration in open hypermedia environments*, Hypertext 97 Proceedings, ACM Press, New York, 1997, 1-12.
[Wiil 99]	Wiil, U.; Nürnberg, P.; Leggett, J.; *Hypermedia Research Directions: An Infrastructure Perspective*, ACM Computing Surveys, Symposium on Hypertext and Hypermedia, 1999.
[W3C]	World Wide Web Consortium. http://www.w3.org/
[W3C EC]	W3C Electronic Commerce Interest Group. http://www.w3.org/ECommerce/
[W3C Meta]	W3C Metadata Activity. http://www.w3.org/Metadata/
[W3C Sec]	W3C Security Initiatives. http://www.w3.org/Security/
[W3C SW]	W3C Semantic Web Activity. http://www.w3.org/2001/sw/
[XBase 00]	XML Base (XBase). http://www.w3.org/TR/xmlbase
[XHTML 00]	XHTML 1.0: The Extensible HyperText Markup Language. http://www.w3.org/TR/xhtml1/

[XIF 99]	XML Interchange Format. http://msdn.microsoft.com/repository/technical/xif.asp
[XInclude 00]	XML Inclusions (XInclude). http://www.w3.org/TR/xinclude
[XLink 00]	XML Linking Language (XLink). http://www.w3.org/TR/xlink/
[XMI 98]	XML Metadata Interchange (XMI). http://www.omg.org/pub/docs/ad/98-10-05.pdf
[XML 98]	eXtensible Markup Language Specification. http://www.w3.org/TR/1998/REC-xml-19980210
[XPath 99]	XML Path Language (XPath). http://www.w3.org/TR/xpath
[XPointer 99]	XML Pointer Language (XPointer). http://www.w3.org/TR/xptr
[XSchema 01]	XML Schema Part 1: Structures. http://www.w3.org/TR/xmlschema-1/ XML Schema Part 2: Datatypes. http://www.w3.org/TR/xmlschema-2/
[XSL 00]	eXtensible Stylesheet Language (XSL). http://www.w3.org/TR/xsl/
[XSLT 99]	XSL Transformations (XSLT). http://www.w3.org/TR/xslt
[Yoo 00]	Yoo, J.; Bieber, M.; *A Relationship-based Analysis*, Hypertext 2000 Proceedings, San Antonio, ACM Press, June 2000. http://www.cis.njit.edu/~bieber/pub/ht00-yoo-bieber.pdf