Dynamic Views of SGML Tagged Documents

B. Fraser, J. Roberts
IBM Canada
895 Don Mills Road, Tower 1
North York, Ontario, Canada M3C 1W3
1-416-448-3618
bfraser,robertsj@ca.ibm.com

G. Pianosi
LivePage Corporation
158 University Ave West
Waterloo, Ontario, Canada N2L 3E9
1-519-885-2181
garyp@livepage.com

P.Alencar, D. Cowan,
D. German, L Nova
University of Waterloo
200 University Ave. West
Waterloo, Ontario, Canada N2L 3G1
1-519-888-4690
alencar,dcowan,dmg,
luisnova@csg.uwaterloo.ca

ABSTRACT

Product information is more frequently being delivered as hypertext webs or documents because of the availability of the World-Wide Web and the associated communications infrastructure. However, this type of document with its large number of files and hyperlinks can become very complex and present significant usability problems for the creator, maintainer and user. Because of this complexity it becomes extremely difficult to implement and maintain dynamic views of a document, a supposed advantage of a hyperlinked structure. In this paper we analyze some of the causes for these usability issues, and then describe some approaches that are being used to make significant improvements to this situation.

Keywords

Hyperlinks, SGML, XML, documentation, World-Wide Web, usability, SQL, relational databases, tagging languages, dynamic views

1. INTRODUCTION

The accessibility of the World-Wide Web and the Internet/Intranet has produced a paradigm shift in the way product information can be stored, distributed and presented. Product information, formerly delivered in hardcopy and softcopy books, is now being delivered as hypertext webs or documents tagged with HTML and accessible through a Web browser.

Putting large information libraries into hypertext webs presents many new challenges for document designers and writers. The information itself must be designed for easy and usable online access, a process that involves organizing information into independent, distinct topics. The result is often a web of information containing thousands of files. Navigating, searching, building and maintaining such a large information web can be very difficult without new organizational and access approaches.

At IBM, we have experience creating webs of product information for several VisualAge products that exceed 10,000 files. The size of these webs creates enormous build and maintenance problems for the writers, and corresponding search and navigation issues for the users. To solve these problems, we looked for new approaches for creating, maintaining, and delivering large information webs.

Tagging or markup languages and the underlying storage models are key issues to be examined as they impact the construction, maintenance, delivery and usability of documents distributed over the World-Wide Web. We assess their impact and examine a document model that separates the concerns of content, structure, navigation and presentation and which enhances usability for the creator and maintainer of documents. We then focus on research being performed by the University of Waterloo, LivePage Corporation, and the IBM Canada Laboratory, which is aimed at implementing this model.

The objective of this research project is to investigate and propose models that address the documentation needs of IBM within this new Web and Internet-based world. The proposed documentation model is being prototyped using LivePage Enterprise software [12]. This software stores documents tagged with SGML-compliant languages in SQL databases. The LivePage software also has a number of integrated tools including a server mechanism that supports SQL statements embedded in documents and dynamic presentation of documents on the Web.

Ideally, the information delivery system should provide multiple views into the information for the user depending on their level of expertise or the type of knowledge being requested. Meta-data about the information can be stored in the SGML or XML markup language, or in the database as attributes.

Current prototypes of the documentation system support:

- Automatic dynamic translation of SGML documents into HTML: Documents stored in a database are translated into HTML and served to the Web at the time they are requested. A table of contents (TOC) and context-sensitive navigation buttons are added to the document to provide the user with advanced navigation capabilities.
- Conditional views (dynamic and static): The same documentation is usually presented in different ways to different users depending on the user's interest or qualifications. For example, some documents have "conditional" sections that are displayed only under certain circumstances. "Static" views (e.g. author vs. user views) are password protected, while "dynamic" views (e.g. OS/2 vs. Windows 95 views) may be changed during navigation.

- Structural views: The document appears to be re-organized with a different clustering of topics depending on the user's requirements. For example, the document could be presented in a tool rather than a task view.
- Multiple table of contents (TOCs): a different TOC is generated automatically for each conditional or structural view of a given document.
- "Intelligent" searching: the model provides support for searching where the results are organized according to a relevance ranking. Searching within a context provided by SGML tags is also supported. Searching is restricted to the current view of the document.
- Unicode documents: the model supports documents written in Unicode allowing the display of documents written in other national languages.

Our goal for this research project is twofold. We wish to enable users of IBM products to navigate and search easily through large databases of product information, which will be dynamic and customizable and may be distributed across an intranet or the Internet. In addition, we wish to simplify the construction and maintenance of these databases so that the creators of product information may concentrate on content and usability instead of issues related to managing large files.

2. MAINTAINING/USING HYPERLINKED DOCUMENTS

Documents created for delivery using the World-Wide Web normally use a number of hyperlinked files tagged with HTML. As these documents grow they become extremely difficult to manage and maintain because of the number of files and links. Maintaining a large Web site is similar to and as complex a task as maintaining a large software system.

Usability of the document is affected from the perspective of both the maintainer and the user. Because of the inherent complexity of the document the maintainer can not easily or correctly modify or add features. Thus, features that may enhance usability are not implemented in a timely manner or often do not work. Further, the user will find it almost impossible to augment the documentation, a practice, which is becoming more common as software systems are constructed from components. Unless these problems are rectified, then the day will come when there will be a requirement for meta-documentation, that is documentation to describe the documentation.

A typical hyperlinked HTML document is constructed from individual text files formed through hyperlinks into a large web or network of pages. There is usually some implied hierarchical structure resembling a table of contents. This structure is usually implemented as a set of hyperlinks or table of contents (TOC) radiating from the main or home page. In addition, the hyperlinks are used to connect related concepts together into a knowledge structure. There are a number of issues to be addressed in adding material or modifying material in such a document. Tasks such as relating content to a file name, adding or moving a page within the structure all become quite complex.

Modifying existing Web pages is quite difficult. How do we find the correct context in a large Web space consisting of files with simple names? It is very difficult to map the contents of substantial text files into names. Adding new pages is also a problem. The page or file name must be added to the existing table of contents; this operation may cause a series of update operations. Deleting a page is similarly complex, not only must the page be removed from the appropriate tables of contents, but all hyperlinks that use this page as a destination from the TOC or as part of the knowledge structure must be found and modified. The problems described here only relate to a single view of a document; the problems increase dramatically when multiple TOCs must be maintained because dynamic and conditional views of a document are required.

Because current Web-based documentation systems rely on files as the storage model, there is little underlying additional structure or language to support features such as security, a history or an audit trail.

Maintenance of a documentation system is an expensive operation that is labor-intensive and requires a large investment in highly paid personnel. Many Web sites become stale or out of date because of this requirement for investment.

Searching a Web document is also a complex operation. An index of all the important words and terms in the set of Web pages must be compiled, so that a local search can be performed.

So far we have discussed only a single author, maintainer or user. How can a document be shared so that the integrity of the document is maintained? In other words how do we prevent multiple authors/maintainers from modifying a document at the same time? How do we ensure that all changes are backed up, so we can recover from disasters or errors?

The tagging language HTML is also a problem; it is not easily extensible. The developer of the Web browser fixes these languages and merges both the structural and presentation aspects of the language into a single tag thereby removing flexibility.

Most of the issues illustrated in this section are related to separation of concerns and extensibility, issues that are usually addressed in software engineering, but only beginning to be examined in the context of document systems [8],[10],[5],[15],[6]. Most tools to support authoring and maintenance of HTML documents focus on the file as the main entity. The authors of these tools pay almost no attention to the fact that these types of documents are just a visual representation of more abstract entities that are interrelated, and the modification of one entity might require changes in another.

Because files are just simple lists with no underlying structure and associated programming language, they are not easily extended to include new structures. The markup language for describing the document is similarly limited and not extensible, and describes a mixture of structure, navigation and appearance. The following list categorizes some of the common problems of HTML documents that arise from these considerations:

- Consistency. Since every page is independent, information common to several pages has to be repeated, presenting a potential consistency problem.[1]

[1] Some HTTP servers support "server-side includes", which are directives to the server that are replaced by the contents of a given file or the result of an executable program. This technique only partially addresses the problem of consistency at a high

- Organization or structure. The organization of an HTML document is bound when the files are created. Restructuring is expensive and requires splitting, collating, renaming or deleting files, and also updating links among files.
- Navigation. Since the navigational structure is embedded in the files, it is difficult to find and modify navigational links.
- Presentation. Information is not isolated from its presentation.[2] If the presentation of pages needs to change, that information has to be changed on a file-by-file basis.
- Referential Integrity. When the name, hyperlink or Uniform Resource Locator (URL) [3] of a resource changes, all the links pointing to that resource have to be updated. This problem exists at both the global level, where the author does not have control over the documents pointing to the local information from an external site, and at the local level.
- Extensibility. Files do not have the associated tools and structures that make them easy to extend their capabilities. Similarly the tagging languages are limited in scope.

The apparent simplicity of the tagging systems and the file structures being used for document systems has a substantial cost. The author of a document mixes together content, structural, presentation and navigation information indiscriminately, and then stores the result in multiple files. Because of this integration, it is difficult to change a document's presentation, structure, and content without a complete rewrite. In addition, because there is no clear distinction among these disparate concepts, a document designer often considers the navigational and organizational structure first, followed by presentation, and finally, if there is any time left, the content. Ideally, this process should be reversed: write the content, define the presentation, and then impose a navigational structure.

Many of the issues described can be resolved with database systems and more powerful tagging languages. In the remainder of this paper we briefly describe extensible tagging languages, and then describe in general terms how databases can be used to address many of the issues associated with creation, maintenance, and use of text-based documentation systems. Finally we present one specific solution using relational or SQL databases.

3. TAGGING LANGUAGES

Tagging languages can be divided into general-purpose and domain-dependent schemes. In the former, the tags have a generic meaning; an H1 tag in HTML represents the biggest type of title tag available. However, this type of tag does not specify whether the tag is a title of a book, the title of a poem, or the name of a country. The only information specified is that the text enclosed by the tag is sufficiently important to render it with a large font and in bold. Domain-specific languages are created to reflect the structure of the document: a book is divided into chapters and the chapters into sections. Each one can have a title. Paragraphs, quotations, citations, foreign words, each can be separately marked with a specific tag type.

Generic languages such as HTML have been extensively used because they are easier to implement since the browser knows the language being rendered for presentation. SGML [7] and its newer derivative XML [17] are metalanguages in that they are used to define domain-specific tagging languages. These meta-languages with an accompanying grammar or document type definition (DTD) [7], define the language. In other words they define the tag types and their use. The accompanying style languages such as XSL [14],[18] and DSSSL [2] provide the flexibility to expand the syntax and semantics of documentation systems and support the separation of the presentation of the document from its content, structure and navigation.

Including more information about the structure of a document benefits both the authoring and browsing processes. The author does not have to make style decisions such as the size of the header for a particular part of the text or whether quotations should be displayed with underscores or italics. Instead, the author can concentrate on the content and the proper use of tags. In a subsequent step, the document can be translated into a generic tag language --such as HTML-- or it can be viewed by using a more sophisticated browser that takes as input the DTD, a style-sheet that specifies how to render each tag in the language and the document. The way the document is displayed can then be easily changed, either by the designer or by the reader. Furthermore, searching can be more precise, since the search engine can take advantage of the structure of the document.

4. DATABASES FOR STORAGE OF TEXT

Both relational and object databases can be used to store the objects found in documentation. The objects are segments of text and the objects to which they refer, such as pictures, graphics, audio or video clips and programs. What are the advantages of using a database over the use of multiple files?

A database can be viewed as a single file containing various structures or meta-data relating all the objects or tables. Extra structures can be imposed upon the database to capture relationships that are not inherent in the document. Although such structures can also be added to files, the languages and tools to support these structures must also be constructed, whereas languages such as SQL, and supporting tools are an inherent part of a database system. We mention some specific examples of useful structures in the following paragraphs.

Almost all hyperlinks in a document fall into four categories, structural or organizational, circular, star, and index. Structural links are used to organize a document into a hierarchy resembling a book. Circular, star and index links form knowledge structures. Circular and star links connect similar concepts together. Circular links form a chain, while star links point to a common destination or concept. Index links are the reverse of star links, in that they are usually presented as part of an index table and connect the concept in the table to all its references. Since hyperlink structures can be categorized, the relationship among hyperlinks can be captured in database structures. This structure can then be scanned to detect missing destinations, or reflect deleted or additional hyperlinks. Multiple organizational structures can also be maintained supporting different views of a document.

A history relationship could be included in the database that included comments about modifications to the data or the date of

---

performance price, since the directive is executed every time the node is requested.

[2] Cascading Style Sheets are trying to solve this problem partially at the file level.

the most recent changes. Such a structure would allow an author or user to find and track the most recent changes to a document.

The aforementioned structures can be used to separate organization and navigation from content, thereby addressing issues related to separation of concerns. Such structures also make it easy to support referential integrity and consistency. In effect the ability to add structures with supporting languages such as SQL or embed those languages in other programming languages makes a highly extensible open environment.

Databases support common functions such as security, backup and replication. The security model in most databases has been well tested over time. Security can be used to lock database structures in order to implement check-in and check-out facilities. Such techniques are essential in managing author interaction in a multi-author document. Backup structures are also inherent in most database systems and are essential in implementing disaster control. Replication facilities support distribution of information across computing platforms.

Storing and manipulating text containing a markup language such as SGML or XML is only valuable if the text can be delivered in manageable chunks similar to those provided by a traditional Web-based system. In the next section we describe the LivePage system and illustrate how it uses database technology to store and deliver Web pages.

5. THE LIVEPAGE SYSTEM

In a previous section, we propose that database systems are an appropriate storage model for tagged text, and support the separation of content, organization, and navigation. We have developed an implementation [12],[19] of this model that satisfies most of the specified requirements, and uses a relational database system. Figure 1 illustrates a simplified view of this implementation; details are presented in the next subsections.

5.1 The Base Implementation

In this implementation we initially embed text, hyperlinks and references to objects such as pictures or audio clips into a single document, but distinguish or "separate" them by tagging conventions using an SGML-compliant tagging language. Thus, if we move to a different entity storage model, we will be able to separate the three types of information easily. Once the document is complete, we verify it against a grammar or document type definition (DTD) before loading the document into a single relational (SQL) database.

Each fundamental tagged structure[3] or document component is loaded into a single field in a relational database table and given a unique identifier. There are three separate tables for the text, objects and hyperlinks. In addition, every word in the document is also placed in an index to facilitate searching when the database is browsed. Optionally, every structure tag can be placed in the same index to facilitate searching for words within specific tagged structures. An author can also create specialized indices by defining the words and phrases that form its content. The various objects such as graphics, video and sound, and links to external programs are stored directly in the object table as "blobs" with the appropriate attributes. The tables for the tagged text, hyperlinks, objects and index are identified in Figure 1, and collectively are labeled as a multimedia database.

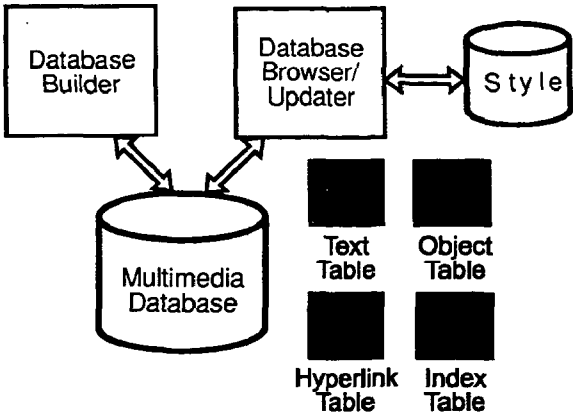

Figure 1 The base implementation

5.1.1 The Toolkit

The basic LivePAGE toolkit is provided to support creation (the database builder), maintenance (the updater), and browse and query (the browser) of the database. Builder functions, namely the verification against a DTD and creation of the database tables are described earlier. We consider documents as trees [13], and the updater allows a substructure (subtree) to be moved, deleted, modified or replaced within the document. While this substructure is being changed, the corresponding section of the database can be locked in order to maintain database integrity. Since we use SQL database technology, the database can be created, updated, and browsed using SQL statements. However, the LivePAGE tools provide an interface that makes the application of the SQL statements transparent.

The database builder and updater are primarily tools for the database administrator or author, while the browser is a tool for the general user to examine the database. The browser supports linear browsing, following hyperlinks forward and backward, and activating objects such as audio and video clips, or links to external programs. The functions of the database builder and updater are presented through a uniform user interface so documents can be created and modified seamlessly.

The browser also supports queries. The queries can be Boolean or simple where the results of the simple query are presented in relevance order with the most relevant result presented first. The simple query can be an English sentence.

Text stored in the database can be extracted and modified using most commercially available structured text editors. Similarly objects stored in the database can be extracted using the updater and can be created or modified using appropriate authoring tools.

5.1.2 Presentation Styles

Tools such as the browser and updater allow the client to view the document stored in the database. However, the document contains only content, structural and navigation information. Presentation or style information is contained in a separate style database and is loaded into the browser or updater when it is invoked. The style database is indicated in Figure 1.

[3] With certain exceptions a fundamental tagged structure contains no tagged substructures.

5.2 Connecting to the World-Wide Web

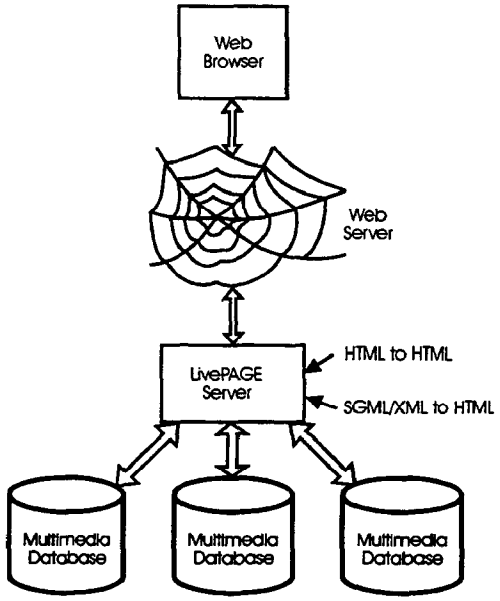

Figure 2 Accessing the database over the WWW

The base implementation described previously allows local access to documents but does not support access through the World Wide Web (WWW). The LivePAGE toolkit contains two other mechanisms for this purpose.

5.2.1 The Server - Producing a Dynamic WWW Site

Figure 2 illustrates the architecture of a dynamic distributed document database system accessible over the WWW. Users accessing a WWW browser such as Netscape or Microsoft Explorer request a WWW page from a WWW server. Using the CGI, NSAPI or Microsoft ISAPI protocol, the WWW server passes the request to the LivePAGE Server module that then accesses a database of WWW pages tagged with SGML or XML. The specific WWW page that was requested is retrieved from the database and then transformed into HTML by the LivePAGE Server, and returned through the WWW server to the WWW browser for presentation.

The transformation shown in Figure 2 proceeds in two steps. First the SGML or XML document is transformed into HTML. After this transformation the HTML is augmented with extra HTML and JavaScript to achieve a specific appearance. A typical presentation would contain a "Table of Contents" (TOC) and navigation buttons. The automatic generation of a TOC and the navigation buttons relieves the author of the WWW site of creating navigational aids, and allows users to orient themselves by returning to the TOC whenever they feel "lost in hyperspace.[4]" The SGML/XML to HTML transformation is a style and so is specified in the XSL style language.

The SGML/XML pages in the database can also contain embedded SQL commands that support access to structured relational databases. The LivePAGE Server intercepts these embedded SQL statements and passes them to the structured database. Thus we can combine multimedia and structured data in a single page.

The LivePAGE Server module is not restricted to a single database, but can retrieve and search WWW pages from multiple tagged document databases as illustrated in Figure 2.

5.2.2 The Publisher - Producing a Static WWW Site

The Publisher is a tool similar to the Server Administrator except that the Publisher generates a static Web site. The generation process creates a WWW site from the database at a specific point in time. If the database changes then the "publishing" process must be repeated.

5.3 Importing Legacy WWW Sites

There are many existing WWW sites that could benefit from the support of document databases. The LivePAGE toolkit provides a facility to import an existing WWW site, to attempt the correction of structural errors, and then to build a tagged document and multimedia database. Once the database is constructed, all the tools previously described can be used.

5.4 Extensibility

The extensibility of the database and tagging approach to documentation systems is illustrated through features that have been added to the database through the LivePAGE toolkit.

We have often augmented a DTD with tags to support features within a document. For example, a document could contain a "rating" tag that is not visible but that describes the intended audience for the document.

The database can be easily extended to provide new features in a document. For example, a history indicating dates on which a specific section of a document has been changed or comments describing those changes can be easily included.

We can also easily provide "views" of a document through the realm concept. A realm is a set of pages in the database with a user assigned to one or more realms through a database table. Thus a user can be excluded from browsing a specific set of pages by being excluded from a realm. Realms can be used to implement security, or tables of contents for limited viewing.

We consider documents as composed of uniquely identified components that are organized into trees. Thus, database tables can specify the table of contents (TOC) or structure of a document. If we wish to create another document from the same set of components then we only need to create new database tables or TOCs to specify the new structure. Thus when a user switches contexts the system is only required to switch to a new TOC. This feature has been added to the original system in order to provide multiple structural views of the same contents.

6. CONCLUSIONS

In this paper we discuss issues related to creating and maintaining large hyperlinked documents, a form of documentation that is becoming more common as the World-Wide Web becomes increasingly pervasive. Specifically we focus on how the separation of content, structure, navigation and presentation can assist in simplifying the maintenance of such documentation and improve usability for both the maintainer and user of the document. This paper illustrates how tagging languages can be

[4] The position in the TOC is highlighted to assist in the orientation process.

used to separate presentation from content, structure and navigation. We then describe how database structures could be used to separate content, structure and navigation from each other.

We also claim that the database approach overcomes significant limitations of Web-based documentation systems related to openness, extensibility, consistency, and referential integrity. Finally we present an approach to documentation systems based on storing text marked with an SGML or XML compliant tagging language in a relational database. Style languages such as XSL are used to transform the SGML into HTML for presentation using the Web. The implementation is open in that it is based on standards in both tagging languages and databases. Further these two standards are extensible.

There are other implementations [1],[4] that use a database specialized to the storage and retrieval of tagged documents. However, these are proprietary in nature and not easily extensible by the user of the system. Other hypermedia systems [9] have been adapted to generate HTML tagged text. However the information does not reside in a database. Other authors have described using relations to represent hyperlinks [11],[10],[16].

Besides the technical advantages, adopting our approach also leads to benefits from a corporate perspective. The approach is cost-effective in terms of leveraging an organization's existing investment and expertise in RDBMS. There is also an ample supply of third-party (or outside) assistance. Because it is based on open standards, developers are not locked into a proprietary solution. In addition, text and data can be accessed using SQL across different vendors, and systems are easily portable to other vendors' systems. Furthermore, since relational database technology is a current *de facto* standard, relational databases are portable across different hardware vendors.

Finally, the database support of our approach to documentation is scalable, flexible, extensible and proven. It scales from stand-alone PCs to mainframes. There is also a wide range of vendors with excellent track records for relational databases. In particular, relational databases have proven the test of time with respect to security, performance, and data replication.

7. ACKNOWLEDGEMENTS

The authors wish to thank IBM Canada for support.

8. ADDRESS

J. Roberts - 1150 Eglinton Avenue East, North York, Ontario, Canada, M3C 1H7 – 1-416-448-3881

9. REFERENCES

[1] ActiveSystems Inc., "ActiveSystems, Reference Manual," 1996.

[2] Sharon Adler, "ISO/IEC DIS 10179.2:1994. Information Technology - Text and Office Systems - Document Style Semantics and Specification Language (DSSSL)," International Organization for Standardization, 1994.

[3] T. Berners-Lee and Masinter and M. McCahill, "Uniform Resource Locators (URL)," Request for Comments 1738, December, 1994

[4] EBT International, "DynaBase Reference Manual," 1996.

[5] F. Garzotto and P. Paolini and D. Schwabe, "HDM - A Model-Based Approach to Hypertext Application Design," ACM Transactions on Information Systems, Volume 11, Number 1, pp 1--26, 1993.

[6] Daniel M. German and D.D. Cowan and P. Alencar, "A framework for formal design of hypertext applications," 4th Brazilian Symposium on Multimedia and Hypermedia Systems, Rio de Janeiro - Brazil, to appear.

[7] C. Goldfarb, "SGML Handbook," Oxford University Press, 1990.

[8] F. Halasz and M. Schwartz, "The Dexter Hypertext Reference Model," Communications of the ACM, Volume 37, Number 2, pp 30-39, February 1994.

[9] G. Hill and W. Hall and D. Roure and L. Carr, "Applying Open Hypertext Principles to the WWW," Proceedings of the International Workshop on Hypermedia Design (IWHD'95), June, 1995.

[10] T. Isakowitz, A. Stohr and P. Balasubramanian, "RMM: A Methodology for Structured Hypermedia Design," Communications of the ACM, Volume 38, Number 8, pp34-44, August, 1995.

[11] D. Lange, "An Object-Oriented Design Method for Hypermedia Information Systems," Proceedings of the 28th Hawaii International Conference on System Sciences, January, 1995.

[12] LivePage Corporation, 158 University Avenue West, Waterloo, Ontario, "LivePage Enterprise," 1999.

[13] E. Mackie and J. Zobel, "Retrieval of Tree-structured Data from Disc," Databases '92, Third Australian Database Conference, Melbourne, February 1992.

[14] Microsoft Corporation, "XSL Tutorial," www.microsoft.com/xml/xsl/tutorial/tutorial.asp, 1998.

[15] D. Schwabe and G. Rossi, "The Object-Oriented Hypermedia Design Model," Communications of the ACM, Volume 38, Number 8, pp 45-46, Aug, 1995.

[16] D. Schwabe, G. Rossi and S.D.J. Barbosa, "Systematic Hypermedia Application Design with OOHDM," Proceedings of Hypertext 96, pp116-118, 1996.

[17] W3 Consortium, www.w3.org/TR/PR-xml-971208, "Extensible Markup Language (XML)," 1997.

[18] W3 Consortium, "A Proposal for XSL," w3c.org/TR/NOTE-XSL.html, 1998.

[19] J. Zobel and R. Wilkinson and E. Mackie and J. Thom and R. Sacks-Davis and A. Kent and M. Fuller, "An architecture for hyperbase systems," 1st Australian Multi-Media Communications Applications and Technology Workshop, Sydney, July 1991.