LivePAGE - A multimedia database system to support World-Wide Web development

Donald D. Cowan
Department of Computer Science
University of Waterloo
Waterloo ON N2L 3G1
E-mail dcowan@csg.uwaterloo.ca

Daniel M. German
Department of Computer Science
University of Waterloo
Waterloo ON N2L 3G1
E-mail dmg@csg.uwaterloo.ca

Eric Mackie
Inforium Technologies Inc.
158 University Ave. W.
Waterloo ON N2L 3E9
E-mail eric@inforium.com

Abstract

The rampant growth of the World-Wide Web (WWW) is largely a consequence of its simplicity. A typical person can quickly learn HTML and start creating WWW pages in an afternoon. As WWW sites become larger and more complex, this inherent simplicity causes multiple problems as many of the current tools and techniques are stretched to address issues they were not designed to handle. These problems are compounded by the rapid proliferation of solutions which are often fairly ad-hoc in nature. In this paper we present a layered model which through its tools and techniques provide a more disciplined approach to constructing and maintaining a WWW site. We then describe an implementation of this model which is based on two more mature technologies; SGML and relational database systems.

Keywords Architecture, World-Wide Web, multimedia, SGML, Web site development, document database, hypermedia.

1 Introduction

The accessibility of simple-to-use, but powerful interfaces or browsers, and the apparent simplicity of HTML has prompted the development of literally millions of hypermedia World-Wide Web (WWW) application sites. In many of these applications both the size and number of the source documents are sufficiently small that each application can be developed and maintained by a single person without the assistance of any methodology or tools (except for a text editor). In contrast, the developer of a large WWW site is constantly struggling to master the complexity involved in the design, development and maintenance of such a site with scores of pages and links.

The WWW is a large collection of interrelated resources, linked through WWW pages tagged using HTML (HyperText Markup Language [3]). HTML pages can point to other resources on the WWW, such as images, video, sound and text. Each resource on the WWW has, at least, one address, known as a URL (Uniform Resource Locator, see [4]). URLs are strongly tied to the file system of the machine in which they reside, and often rely on common differences between file systems such as case-sensitivity and the length of file names. This reliance makes portability of documents between machines difficult.

The WWW is quite relaxed about the type of its components. Although HTML is defined by an SGML DTD,[1] an HTML document rarely complies with this definition.[2] Consequently, a WWW server normally relies solely on the extension to a file's name to determine its type.

Because of the focus on the markup language HTML, most of the tools produced to date are oriented toward editing HTML files. Very little research has been directed toward authoring systems that manipulate WWW sites as a collection of nodes and links, and that view HTML more as a presentation language than as a storage format.

The WWW is thriving mainly because of its simplicity. A typical person can quickly learn

Proceedings of the Second Australian Document Computing Symposium, Melbourne, Australia, April 5, 1997.

[1] See [10] for a description of SGML and DTDs

[2] Bray found that only 4.9% percent of 3 million HTML documents he analyzed have a <!DOCTYPE declaration (which is required in a SGML document). It is unknown what percentage of these were actually compliant with any of the HTML DTDs [6].

HTML and start creating WWW pages in an afternoon. HTML was meant to be a generic structural language; but power users felt it was too restrictive, especially for presentation purposes. Since its appearance, HTML has been expanded to support more and more presentation features including executable code (Java and JavaScript) and style-sheet information (Cascading Style Sheets –CSS [5]–). These new features while enriching the capabilities of the WWW are making WWW documents more and more difficult to read, write, and maintain.

The W3 Consortium (the body in charge of standardizing and promoting the WWW[3]) has always claimed that simplicity is not an objective for any new WWW standards, because tools can isolate the user from such complexities. For instance, HTML files with "frames" and "style-sheets" might be difficult to read, but a tool such as a WYSIWYG editor can be used to create and display them in a more readable format.

Most tools to support authoring of WWW applications have been focusing on the HTML file as the main entity. The authors of these tools pay almost no attention to the fact that HTML pages are just a visual representation of more abstract entities and that, in a WWW site, entities are interrelated, and the modification of one entity might require changes in another. The following list highlights some of the common problems that are a consequence of using the file as the only medium for the WWW hypermedia system:

- Consistency. Since every page is independent, information common to several pages has to be repeated, presenting a potential consistency problem.[4]

- Organization. The organization of a WWW site is bound at the time that the files are created. Restructuring is expensive and requires not only splitting, collating, renaming or deleting files, but updating links among files.

- Navigation. Since the navigational structure is embedded in the files, it is difficult to find and modify.

- Presentation. Information is not isolated from its presentation.[5] If the presentation of a set of pages needs to change, that information has to be changed on a file-by-file basis.

- Referential Integrity. When the URL of a resource changes, all the links pointing to it have to be updated. This problem is common both, at 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.

The apparent simplicity of HTML has a substantial cost. The author mixes together content, presentation and navigation information indiscriminately, and then stores the result in single file. Because of this integration, it is difficult to change a WWW site's presentation, structure, and content without a complete rewrite. In addition, because there is no clear distinction among these disparate concepts, a WWW site designer often considers the navigational 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.

2 WWW development systems

Different approaches have been tried to separate the content from its organization and presentation. We highlight some of them in this section.

2.1 Other tagging languages

Since HTML has limited structural components and is more oriented toward presentation, some authors have chosen to use another tagging language for the "master" of the information they maintain. The most common tagging system chosen uses SGML-compliant tagging languages which have a richer structure than HTML. These languages allow the author to characterize the structure of a set of documents, and to enforce this structure. Using ad-hoc filters, the SGML files can be converted to HTML. The advantage of such an approach is that the structure of the published information is kept separate from its presentation. If the user decides to change the master document's appearance, then only the filters need to be changed. A number of companies who publish the same information in various forms such as HTML, paper, PostScript or Acrobat, have chosen SGML-compliant tagging languages as their master format.[6] Other master formats have been proposed: LaTeX, word-processor based tags, and RTF. All these consider a WWW site as a text document, with well specified rules to translate it into HTML. Properly used, this

[3] http://www.w3.org

[4] 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 only partially addresses the problem of consistency at a high performance price, since the directive is executed every time the node is requested.

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

[6] At the Developer's Workshop "Using SGML on the World Wide Web", during the 5th International WWW Conference, Sun Microsystems and Novell Inc. acknowledged that they have taken this approach to deal with the complexity of publishing their manuals.

approach avoids inconsistencies, and separates presentation and navigation information from content. The main disadvantage of this approach is its focus on documents with a linear structure, such as books and articles, rather than the highly interconnected structures of objects typical in hypermedia applications.

2.2 Publishing SGML-compliant documents directly on the WWW

Some WWW publishers make their documents only available for users with specific SGML browsers (such as Grif Symposia [16] and SoftQuad's Panorama). They publish documents tagged with SGML-based tagging languages compliant with a standard DTD. A style-sheet is produced for the DTD, and is downloaded with the document to ensure proper rendering. SoftQuad is distributing copies of its browser to promote this concept.[7]

2.3 Macroprocessor-based systems

In this approach the master information is stored in an ad-hoc tagging format, and a preprocessor or macroprocessor tailored specifically for HTML, is used to convert the files into HTML pages. A flexible macroprocessor can greatly assist in WWW development, reducing inconsistencies and separating content from presentation and navigation structure.[8] We have developed a prototype macroprocessor system using the M4 preprocessor.

2.4 Page image systems

PDF and PostScript are popular formats used to publish information on the WWW. Both require special browsers to provide full control of the document typography.

2.5 Hypermedia-based systems

The success of the WWW has prompted the authors of some hypermedia systems such as Microcosm to adapt them to generate HTML pages. Hill et al. proposed to unify Microcosm[9] and the WWW in three ways [11]:

- Microcosm-aware WWW clients, that is enhancing clients to support Microcosm primitives.
- Generating static pages out of a Microcosm system.
- Using CGI scripts to allow the interaction of a Microcosm server with typical WWW clients. The CGI scripts would convert Microcosm requests to HTML browsable files.

In all these cases, the information does not reside on a database.

3 Modeling a WWW site

The spectacular growth of the WWW has led to a corresponding growth in ad-hoc solutions related to specific WWW problems, some of which are highlighted earlier. Categorising the information elements that constitute a WWW site and the operations performed on these elements should suggest a model that could lead to generic approaches for solving many of these problems.

The three basic types of information involved in a hypermedia system such as the WWW are: text, objects and hyperlinks. Text and objects are the fundamental units that are presented in WWW documents, while hyperlinks provide intra- and inter-document navigation.

3.1 Text

Text is ubiquitous in the WWW. Many hypermedia systems consider text only as a string of characters, but recent advances in the characterization of text have shown that structural information contained in markup languages such as those derived from SGML [10], is a fundamental component of any text-based document.

Each text element such as a paragraph or heading is tagged with an SGML tag to comply with a particular document structure or DTD.[9] Text and its associated structure should be stored separately from information about its appearance, so that the presentation can be tailored to specific environments or situations.

3.2 Objects

The remaining types of information contained in a WWW document such as images, video clips, and Java scripts can be characterized as objects. For example, consider images of paintings, which might be part of an on-line Museum on the WWW. In this particular scenario, an abstract class `image` could have three main methods: get_thumbnail, get_gif, and get_jpeg. However, the images we have are in tiff and PostScript format. A feasible solution is to create two subclasses for gif and jpeg, and then the translations from tiff and postscript format could be described within the methods of the object. For instance, the object `image` could have methods `low_resolution`, `thumbnail`,

[7] See http://www.ncsa.uiuc.edu/SDG/Software/Mosaic-/WebSGML.html

[8] We used this approach to create the WWW version of the "The University of Waterloo Undergraduate Calendar," which is composed of around 500 pages. The savings in development time were enormous compared to using plain HTML files.

[9] Document Type Definition, which formally describes the structure of a class of SGML documents.

`black_and_white` which are overwritten by the methods of its subclasses. Figure 1 shows the OMT diagram of the image and GIF objects with an example of their corresponding methods.

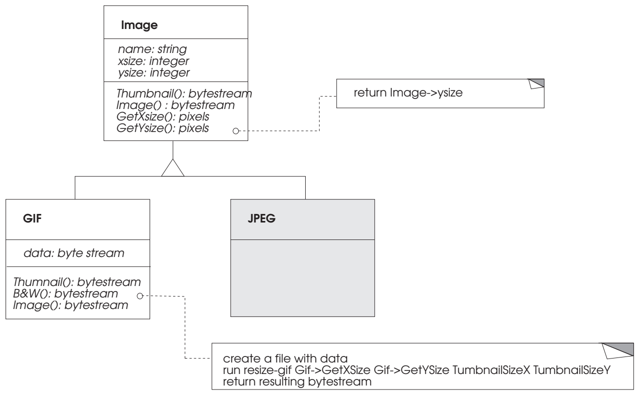


Figure 1: OMT diagram of the Image superclass

3.3 Hyperlinks

Hyperlinks are used for navigation around a WWW site, where a text segment or graphic is activated to link to a new location. Relations can be used to represent hyperlinks since the entity-relationship model has been used successfully by several authors in hypermedia system design [14, 13, 17].

3.4 Authoring Tools

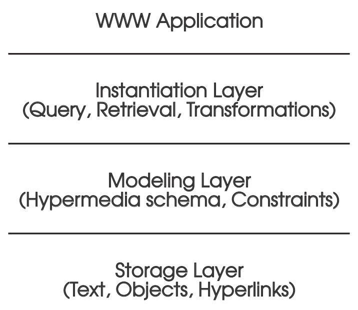
The three types of entities require different tools to support creation and maintenance. For example, we can use a grammar-driven editor to process tagged text and different tools to manipulate the subtypes of objects such as sound, video or graphics.

3.5 Persistence

Text, objects and hyperlinks are persistent entities that can be modified but must last in some form for the lifetime of a WWW document. All these entities can be stored as separate files, but using databases with their many advantages for storage and retrieval is a more desirable approach.

3.6 A layered model

Since there are three distinct types of entities, and operations on these entities form groups, a layered model is a good approach to describing the relationships between entities and operations needed to construct a WWW site. Figure 2 is a diagram of the layers showing the entities in the storage layer, the operations forming the next two layers, and the completed WWW application as the final layer. This layered model is useful for describing and categorizing the various aspects of the WWW site, but the actual implementation may differ somewhat from this "ideal."



WWW Application
Instantiation Layer
(Query, Retrieval, Transformations)
Modeling Layer
(Hypermedia schema, Constraints)
Storage Layer
(Text, Objects, Hyperlinks)

Figure 2: Layers in a hypermedia system

The storage layer

The storage layer contains the three types of entities which are stored in an appropriate database system.

The modeling layer

The modeling layer acts as the "glue" to create a unified hypermedia database. The modeling layer incorporates the following metadata:

- A hypermedia schema which is a description of the types of structures inherent in each type of entity. For example, the structure of the text could be described using a collection of DTDs, the objects through a class library, and the hyperlinks through a simple relation schema showing source-destination pairs.

- A set of constraints to maintain consistency across the database. These constraints ensure that if a textual component refers to an object or other textual component that the latter exists and is not removed from the storage layer while still being referenced.

The instantiation layer

Once the information has been retrieved and the navigational structure applied, the document still must be transformed into the appropriate appearance. This function is the responsibility of the instantiation layer which retrieves the necessary information from a repository in the form of instantiation rules and scripts. The instantiation layer includes:

- Functions to retrieve information from the database system.

- Functions to convert queries to entity retrieval commands.

- Transformation definitions. Transformations are intended to be applied to the stored entities, often in response to queries.

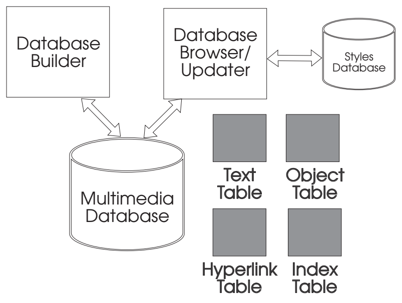


Figure 3: The base implementation

  - text entities can be processed using an SGML processing language, such as Omnimark [7] or Jade (a DSSSL engine). For example, a number of memoranda documents tagged with SGML could be transformed by extracting the name of the author, the recipient, the date, and the first 50 characters of the text of the memo, and then presenting the result in HTML.
  - For presentation purposes objects might require extra manipulation such as format conversion.

The instantiation layer could use scripts written in a language such as Perl[18] to describe how to retrieve and transform objects and how to apply the navigational information specified as hyperlinks.

Advantages of the layered model

This layered model has several advantages including:

- A clear separation of content, presentation and navigational information.
- Independence from specific storage structures allowing the designer of a system to make choices based on the available database systems and corresponding authoring tools.
- Consistency between entities ensuring the integrity of the storage model.

4 The LivePage implementation

In Section 3 we propose a layered model to describe WWW applications and to separate the different entities and operations. We have been developing an implementation [19, 12] of this model that satisfies the majority of the specified requirements, and uses a single relational database system. Figure 3 illustrates this implementation; the details are presented in the next sections.

4.1 The base implementation

In this implementation we initially embed text, hyperlinks and references to objects into a single WWW 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 DTD before loading the document into a single relational (SQL) database.

Each fundamental tagged structure[10] 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. 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 various tables are identified in Figure 3.

The toolkit

The LivePAGE toolkit is provided to create (the administrator), maintain (the updater), and browse and query (the browser) the database. Administrator functions, namely the verification against a DTD and creation of the database tables are described earlier in Section 4.1. We consider documents as trees [15], and the updater allows a substructure (subtree) to be removed from the database and modified or replaced. 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 administrator and updater are primarily tools for the database administrator, 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 browser also supports queries. The queries can be Boolean or free-form where the results of the free-form query are presented in relevance order with the most relevant result presented first.

Text stored in the database can be extracted and modified using a structured text editor. Similarly objects stored in the database can be extracted using the updater and can be created or modified using appropriate authoring tools.

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

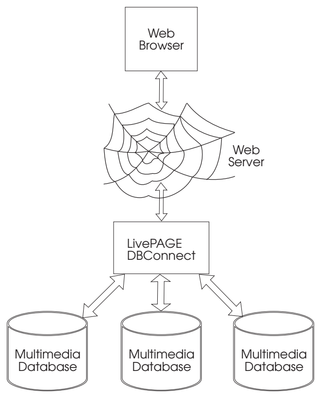


Figure 4: Access to the database over the WWW

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 structural 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 3.

4.2 Connecting to the WWW

The base implementation described in Section 4.1 allows local access to documents but does not support access through the WWW. The LivePAGE toolkit contains two other mechanisms for this purpose.

The Publisher - producing a static WWW site

The Publisher is a tool that can process the database and create a set of WWW documents that can then be stored and retrieved from a WWW site using any of the widely available WWW browsers. In addition to providing the document content including appropriate objects and relations, the Publisher generates a "Table of Contents" (TOC) and navigation buttons as an extra navigational aid. 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."[11]

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. The next section describes a dynamic approach to the same problem.

dbConnect - producing a dynamic WWW site

Figure 4 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 MSAPI protocol, the WWW server passes the request to the LivePAGE dbConnect module that then accesses a database of WWW pages. The specific WWW page that was requested is retrieved from the database and returned through the WWW server to the WWW browser for presentation. The LivePAGE dbConnect module is not restricted to a single database, but can retrieve WWW pages from multiple document databases as illustrated in Figure 4.

4.3 Legacy WWW sites - the Builder

There are many existing WWW sites that could benefit from the support of document databases. The LivePAGE toolkit contains a program called the Builder that can import an existing WWW site, attempt to correct structural errors, produce a TOC, and then build a document database. Once the database is constructed, all the tools previously described can be used.

5 The implementation and model – a discussion

The LivePAGE implementation retains the spirit of the layered model by maintaining the separation between the hypermedia schema and constraints, the transformations, and also the separation among the three types of entities inherent in a WWW site.

The storage model corresponds to the relational database used to handle various tables containing text, objects and hyperlinks. The hypermedia schema and constraints in the modeling layer are controlled through the DTD, where the database is checked against the DTD whenever the database is created or modified. The implementation starts with a single document which is then used to generate a set of relational tables, one for each of the three types of entities. The instantiation layer is also separate in that a style database is used for presentation in both cases. When documents are sent to WWW browsers, the style is implicit in the HTML tags.

There are a number of implementations [1, 8] that use a database structure specialized to the storage and retrieval of tagged documents. We made a practical compromise in our choice of

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

structure in that we chose a single relational database, since most organizations now support the relational model. Based on the length of time taken to adopt relational technology, they are probably a decade away from moving to other databases systems, and so the solution we propose should last a long time. Further, if organizations do switch to other database approaches later, the text, objects and relations are easily separated because of the tags.

The structure of the database used in the LivePAGE model is optimized for retrieval rather than updates, since we expect that a document will be browsed more often than modified. The organization of the internal tables of the database allows a fast retrieval of a subtree of the original document at the expense of relatively slow updates; currently, an update is implemented as a deletion and then an insertion. We are investigating better algorithms to perform updates.

There is approximately a six-fold difference between the size of the original untagged document and its equivalent database. This increase in size is caused by a number of factors including: the system tables, the indexing associated with fast retrieval of information from the database tables, and the indexing of every word in the document (with the exception of stop words).

Currently, LivePAGE is not compatible with the two emerging standards for style-sheet description of SGML documents and HTML (DSSSL [2] and CSS [5] respectively). We expect that this will be changed in future versions of LivePAGE.

The implementation, like the model, supports referential integrity. In the LivePAGE implementation referential integrity is inherent in the tagging since it is simple to verify that all the links and anchors of a document exist.

Access and update control, locking mechanisms, and rollback are all inherent in the LivePAGE implementation, since LivePage uses an SQL relational database as its storage model.

The LivePAGE tools provide extra navigational aids such as Table of Contents generation. Such a facility could also easily be provided in the model.

A complete WWW site is stored in a single database that can be easily moved and does not have file system dependencies.

6 Future Work

LivePage is successful in solving many of the common problems inherent in WWW site development. However, there are a number of issues that need to be addressed.

As more and more information is made available on the WWW it becomes increasingly difficult to find the information that you want. Even search tools which use ranking to return only the results that are the most relevant, are nearing their limits as to how much text they can effectively index. One technique to increase the search effectiveness is to include in the ranking algorithm, information that can be derived from the HTML tags about the structure of the document. Unfortunately, the HTML tagset is not very rich in its ability to describe the structure of all documents accurately. Augmenting HTML with additional SGML tags would be one way to overcome this problem.

Complex WWW sites do not just consist of static textual data with a few added graphics. It is often desirable to include other data, such as a stock market table, that is more dynamic and may be constantly changing. Frequently, this kind of data is also stored in relational databases. We have created a number of demonstrations of how this kind of data can be seamlessly integrated into the WWW site. Further analysis is needed to make this approach comparable to the process of building a simple WWW page.

Even the static textual data that makes up the majority of information in a WWW site is dynamic if looked at over a period of time. Frequently, a WWW site will go through several versions as it is enhanced and maintained. In some applications it is important to keep track of these different versions and to be able to recall an earlier version of the site. This form of version control could be added to the LivePAGE system.

Currently databases created for WWW sites are tagged with HTML, rather than SGML to avoid translating SGML tags to HTML tags and perhaps encountering incompatibilities. However, HTML may not adequately reflect the structure of the information, and SGML may be a better solution. However, it will be necessary to allow the database administrator to define sets of transformation rules to convert SGML into HTML. Such a capability requires further study.

The LivePAGE document database tools only support a single DTD per document, whereas a document may have several different types of structures. This feature is supported by the SGML definition but is not currently available in the LivePAGE system.

7 Conclusions

We have described a system that provides the tools necessary to create and maintain large, complex WWW sites. The system utilizes the power of SGML and relational database systems to solve many of the problems with which developers of large WWW sites are currently struggling. A clear separation is provided between the content, presentation, and navigational structure of the WWW site. This separation allows authors to focus on what they do best, writing content. On large WWW sites there will be other experts to focus on presentation and navigation. Even on smaller WWW sites, where there is only one expert, this separation can provide significant advantages by allowing the author to focus on each aspect of WWW development in turn.

References

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

[2] Sharon Adler (editor). *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 D. Connoly. Hypertext Markup Language – HTML/2.0. Request for Comments 1866, November 1995.

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

[5] Bert Bos, Dave Raggett and Hakon Lie. HTML3 and Style Sheets,W3C Working Draft 10-Jul-1996, July 1996.

[6] Tim Bray. Measuring the Web. In *Proceedings of the 5th International WWW Conference*, pages 993–1005, May, 1996. W3 Consortium, Elsevier.

[7] Exoterica Corporation. *Omnimark Reference Manual version 2 Release 4*. Exoterica Corporation, 1993.

[8] EBT International. *DynaBase Reference Manual*, 1996.

[9] Andrew M. Fountain, Wendy Hall, Ian Heath and Hugh C. Davis. MICROCOSM: An open model for hypermedia with dynamic linking. In *Proceedings of the ECHT'90 European Conference on Hypertext*, Building Hypertext Applications, pages 298–311, 1990.

[10] Charles Goldfarb. *SGML Handbook*. Oxford University Press, 1990. (0-19-853737-9).

[11] G. Hill, W. Hall, D. De Roure and L. Carr. Applying Open Hypertext Principles to the WWW. In *Proceedings of the International Workshop on Hypermedia design IWHD'95*, June 1995.

[12] The Information Atrium Inc., 158 University Avenue West, Waterloo, Ontario. *LivePage Tools*, 1996.

[13] T. Isakowitz, E. A. Stohr and P. Balasubramanian. RMM: A methodology for structured hypermedia design. *Communications of the ACM*, Volume 38, Number 8, pages 34–44, August 1995.

[14] D. Lange. An Object-Oriented Design Method for Hypermedia Information Systems. In *Proceedings of the 28th Hawaii International Conference on System Sciences*, jan 1994.

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

[16] Jean Paoli. Extending the Web's tag set using SGML: Authoring new tags with Grif Symposia. In *Proceedings of the 5th International WWW Conference*, pages 1095–1103, May, 1996. W3 Consortium, Elsevier.

[17] D. Schwabe, G. Rossi and S.D.J. Barbosa. Systematic Hypermedia Application Design with OOHDM. Technical Report 30, Departamento de Informática, Pontifícia Universidade Católica, 1995.

[18] Larry Wall and Randal L. Schwarts. *Programing perl.* O'Reilly & Associates, Inc, 1991.

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