

A Data Model for Educational WWW Servers

Denis Helic

**Institute for Information Processing and Computer Supported New Media (IICM),
Graz University of Technology Graz, Austria
e-mail: dhelic@iicm.edu**

Nick Scherbakov

**Institute for Information Processing and Computer Supported New Media (IICM),
Graz University of Technology Graz, Austria
e-mail: nsherbak@iicm.edu**

Vanessa Mayrhofer

**Institute for Information Processing and Computer Supported New Media (IICM),
Graz University of Technology Graz, Austria
e-mail: vmayr@iicm.edu**

Abstract: Although using HTML as a data exchange format in a client-server system is quite acceptable, and its platform independence has helped HTML to become the de facto standard today, use of HTML as a data storage format for WWW servers deserves examination so that a more useful model can be found.

In this paper we briefly describe a new data model, the HM Data Model, which offers far more advanced structuring capabilities than HTML. We believe that this more sophisticated model offers many advantages with respect to server management and user orientation.

We offer also a Formal method of mapping data structures based on HM-Data Model onto HTML documents.

1. Introduction

It is widely recognized that the Internet offers tremendous potential for education. For example, it is an excellent distribution channel for educational materials and courseware in particular. Normally, the term "dissemination of courseware over the Internet" refers to

an interactive browsing of courseware using systems such as the World-Wide Web (WWW), the most well-known of the interactive information retrieval tools on the Internet.

HyperText Markup Language (HTML) is one of the main concepts upon which the World Wide Web is based. It provides certain capabilities: it provides facilities for describing how information should be displayed, it allows authors to include other forms of data in documents, (e.g. images, sound, pictures, etc.) and it lets them incorporate so-called computer-navigable links into documents. It is a markup language, meaning that it does not allow authors to specify exactly how the text will appear when it is rendered, but rather allows them to specify different basic elements of the text such as the level of heading. How the text is actually rendered depends on the machine it is being displayed on, and may appear different according to what fonts are available on that particular machine. To view WWW HTML documents, one of the many readily-available WWW browsers such as Netscape or Mosaic, which run on practically all hardware platforms, should be installed.

Normally, HTML is not just viewed by users using a client, but also is used to store documents on servers. Here, in contrast to the exchange of documents, it has a number of drawbacks.

Link anchors in HTML are embedded directly in documents, using so-called tags to mark the beginning and end of anchors. Using embedded links poses a number of problems in WWW documents. In particular, let us mention two the most well known problems related to the provision of courseware using WWW:

- First of all, references from one document to another are stored solely within the source document, making it impossible to follow links backwards or to find out what, if any, documents reference a given document. This leads us to the "dangling links" problem: often one must remove documents from WWW servers for one reason or another and the result often ends up being many links pointing to nowhere.
- Also, the lack of structure found in this simple data model is in need of much improvement. Thus, links in WWW are not context-dependent; all links in a document are present all the time. This has two major consequences.
 1. Firstly, the reuse of courseware materials is unsatisfactory, because an author can refer to other documents but cannot embed them locally with possibly new or different hyperlinks.
 1. Secondly, the presence of links completely unrelated to the current context (course) rapidly leads to reader disorientation.

The conclusion that can be drawn here is that although using HTML as a data exchange format in a client-server system is quite acceptable, and its platform independence has helped HTML to become the de facto standard today, use of HTML as a data storage

format for WWW servers deserves examination so that a more useful model can be found.

In this paper we briefly describe a new data model, the HM Data Model, which offers far more advanced structuring capabilities than HTML. We believe that this more sophisticated model offers many advantages with respect to server management and user orientation.

We offer also a Formal method of mapping data structures based on HM-Data Model onto HTML documents.

2. The HM-Data Model

Current trends in logical data modeling can be classified into two main groups:

- Data models supporting links embedded within primitive nodes (i.e. links are local in the sense that they belong to individual documents);
- Data models supporting links as fully-fledged addressable objects (i.e. links are global in the sense that they belong to the hypermedia database as a whole).

In the HM-Data Model, information is not simply stored as HTML documents which have various references between them but rather is organized into units called structured collections (S-collections). An S-collection can be seen as a closed environment which contains a structure of navigable links between so-called Virtual Documents contained in it. Virtual Documents are documents which contain only information and not links. Virtual Documents contained in an S-Collection, are called members henceforth (see Fig. 1).

Thus, the HM-Data Model supports a third type of linking, conceptually somewhere in between the local and global link models. In the HM-Data Model, links neither belong to individual nodes (note that Virtual Documents are free of embedded links), nor they are globally addressable objects, but they are encapsulated within hypermedia containers called S-Collections. By definition, links cannot point outside an S-collection, but only between its members; hence S-collections represent well-defined chunks of information, which may be re-used in various contexts without concern for superfluous hyperlinks. The model compensates for the restriction of links to local contexts by introducing memberwise inclusion of hypermedia chunks and orthogonal browsing semantics to switch local context.

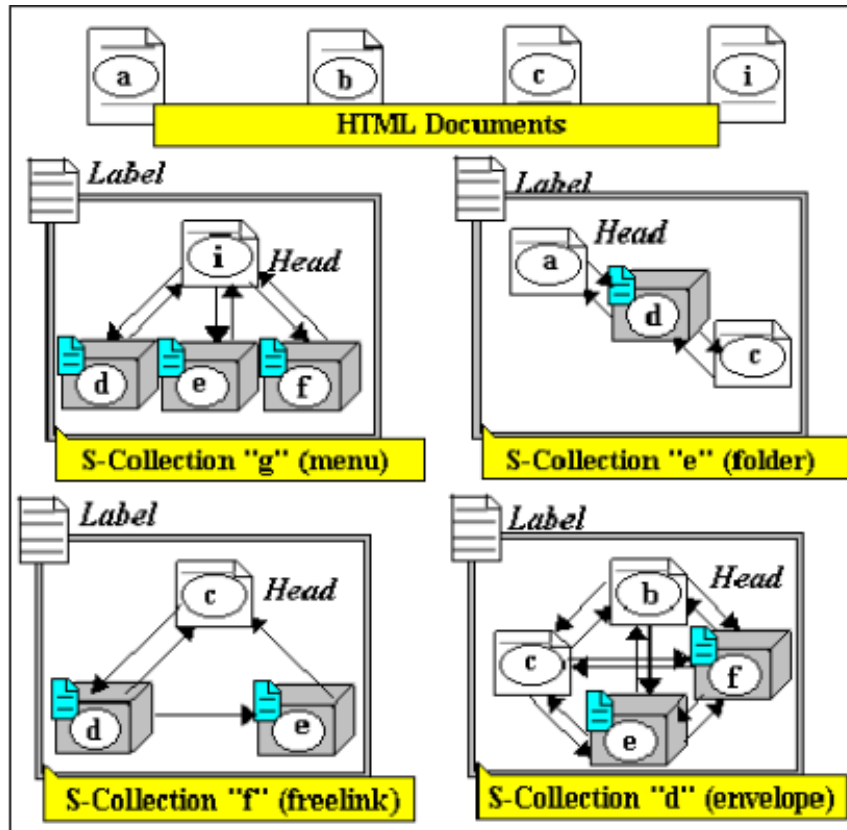


Figure 1: Internal Structure of S-collections

There are several predefined types of S-collection. An envelope has links back and forth between every combination of two members in the collection. A folder joins members into a list, with links to next and previous members. In a menu, one specific member (called the collection head) has links to and from each of the other members. Lastly, with freelinks, authors can link the collection's members in any manner they wish (see Fig. 1). Not only do menus have collection heads, but so does every other type of S-collection. The collection head is a member designated to be seen first when the user starts browsing an S-collection. This, as we shall see, is important with respect to the navigational mechanism in this paradigm. Also, an S-collection can have associated with it a label, a document which describes the content of that collection, and which is the first document seen when the user accesses the collection. These S- collections, which can be seen as separate containers in which only links between members of that container can be seen or followed, can also themselves be members of other S-collections and can have either other S-collections, Virtual Documents or both as members.

The nesting of S-collections can be arbitrarily deep. Recursive membership is possible, i.e. an S- collection can have itself as a member or contain another S-collection which has it as a member. Also, each collection is required to have a unique name which is used to support URL (Uniform Resource Locator) naming scheme. The name is used to access S-collections by means of WWW browsers and for specifying an S-collection for reuse (i.e. having the very same collection appear in two different S-collections and having it linked

differently in each case). The reuse is possible as links are stored separately from virtual documents.

There are three important navigational operations, which are called OPEN, CLOSE and ACCESS. The OPEN operation lets users do just that- when they have navigated to a certain S-collection, have read its label and would like to explore its contents, they open it. As mentioned earlier, each S-collection has a certain internal structure. When one is opened, the environment the user is in is switched from one collection to another. The internal structure of the opened collection is there for the user to browse, so until another collection is opened, the user can only navigate within this collection, always starting with the head of that collection (see Fig. 2).

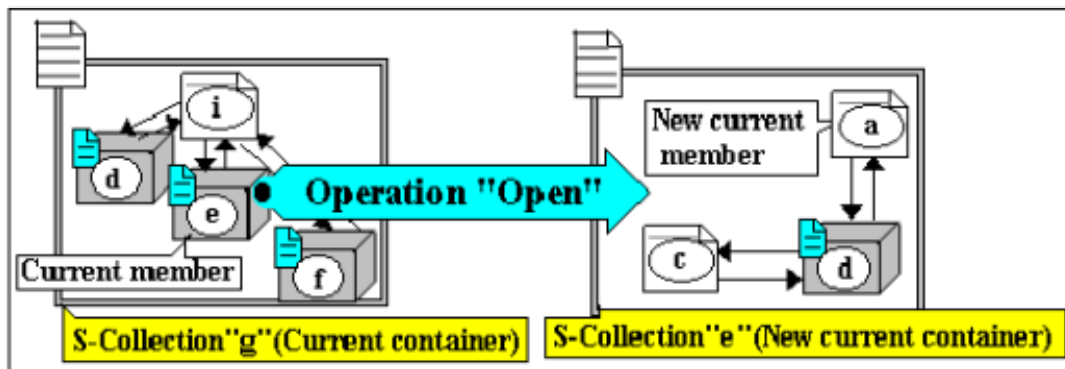


Figure 2: Operation "Open S-Collection"

The CLOSE operation is the opposite of OPEN. It causes the user to leave the browsing environment of one collection and go back to the context of the collection in which it is contained. The third operation, ACCESS, is what the user uses to navigate between the members within a collection. It is similar to following a hyperlink, except that when users access a member of a collection, that member could be a virtual document or it could be another collection, and in this case, what they see first is its label, so they can decide whether to open it or access another member.

3. Projecting HM-Data Model onto the Web

As can be seen above, the HM-Data Model provides better structure and link maintenance than the data model currently used in the Web. If this model were used on Web servers it would save time and effort for server administrators and would also allow them to structure their data in a more advanced manner. It would eliminate the headache of link maintenance as this would be taken care of automatically. It would allow the same documents to contain different links in different contexts, allowing the reuse of information in various contexts. What we are suggesting is a union of the advantages offered by HTML and those of the HM-Data Model.

The idea is to use the advanced HM-Data Model to store information and to use HTML for that which it is well suited, i.e. the Mark Up and exchange of data. All that must be done here is to map the information and structure which is on the server onto elements that are well known in current Web terminology.

Virtual documents are prepared locally by means of an ordinary HTML editor and are stored into a special WWW server. Such virtual documents are further inserted into S-collections by means of a special authoring system running on the server site. S-collections can be accessed by a standard WWW client (for example, Mosaic, MS Explore or Netscape).

Thus, from users point of view, S-collections residing on the server can be treated exactly in the same way as conventional HTML documents. They are provided with URLs, and can be accessed and referred to from within other HTML documents. The only difference is that an S-collection encapsulates a particular navigational which is activated via an additional "Open" operation.

4. Formal mapping of HM-Data Model onto Node-Link model

The following notation is used hereby to refer to navigational operations available in the HM-Data Model:

1. (<S-collection>)

Each S-collection is put into brackets symbolizing a closed navigational system of the S-collection;

2. ->

This symbol denotes the ACCESS operation in HM-Data Model

3.(->

This symbol denotes the OPEN operation in HM-Data Model

4. ^)

This symbol denotes the CLOSE operation in HM-Data Model

The following infer rules can be used by a software component (i.e. Gateway) converting a database structured in accordance with the HM-Data Model, into conventional HTML documents.

1. <Database> ::= <label>(<S-collection>)

2. <S-collection> ::= <folder>

3. <S-collection> ::= <menu>

4. <S-collection> ::= <envelope>
5. <folder> ::= <member₁><member₂>...<member_n>
- 6a. < member₁> ::= < member_n> | < member₂>
- 6b. < member₂> ::= < member₁> | < member₃>
- 6c. < member_n> ::= < member_{n-1}> | < member₁>
7. <menu> ::= <menu_head>
8. <menu_head> ::= <head><menu_member>
9. <menu_member> ::= <member><menu_head>
10. <envelope> ::= <envelope_member>
11. <envelope_member> ::= <member><envelope_member>
12. <head> ::= <label>(<S-collection>) | VirtualDocument
13. <member> ::= <label>(<S-collection>) | VirtualDocument
14. <label> ::= VirtualDocument

5. An Example

Let us consider the following user session. Suppose the user accesses the S-collection "g" (see Fig. 1) using a known URL. The label of the S- collection (i.e. the virtual document) is automatically converted to an actual HTML document by means of embedding additional navigational means (button "Open" in this particular case) , the resultant HTML document is displayed on a client site. Whenever the user "opens" the S- collection "g", head of the S-Collection "g" (i.e. document "i") is automatically converted to an actual HTML document by embedding menu of (i.e. links to) all the members. Suppose, the user activates the link to the member "e" which is in turn a complex collection. The S- collection "e" becomes the current member, and a virtual document defined as its label is displayed with a corresponding link to "I" and an additional anchor "Open". Now the user is able to "open" this S-collection "e". Once the S- collection is "opened" the head (i.e. document "a") enhanced with links encapsulated within the current container becomes available. Now the user may "close" the S- collections "e" and "g".

The sequence of operations generated by the above discussed rules may look as follows:

<Database>

-> <Label>(<S-collection>)

-> VD-Label-g(<S-collection>)

-> VD-Label-g(-> <menu>)

-> VD-Label-g(-> <menu_head>)

-> VD-Label-g(-> <head><menu_member>)

-> VD-Label-g(-> VD-i<menu_member>)

-> VD-Label-g(-> VD-i-> <member><menu_head>)

-> VD-Label-g(-> VD-i-> VD-Label-e(<S-collection>)<menu_head>)

-> VD-Label-g(-> VD-i-> VD-Label-e(-> <folder>)<menu_head>)

-> VD-Label-g(-> VD-i-> VD-Label-e(->
<member_1><member_2>...<member_n>)<menu_head>)

-> VD-Label-g(-> VD-i-> VD-Label-e(-> Vd-
a<member_2>...<member_n>)<menu_head>)

-> VD-Label-g(-> VD-i-> VD-Label-e(-> VD-a^)<menu_head>)

-> VD-Label-g(-> VD-i-> VD-Label-e(-> VD-a^)<head><menu_member>)

-> VD-Label-g(-> VD-i-> VD-Label-e(-> VD-a^)VD-i<menu_member>)

-> VD-Label-g(-> VD-i-> VD-Label-e(-> VD-a^)VD-i^)

6. Conclusion

As can be seen, the data structure which is currently widely used in the Web is not nearly as advanced or useful as it could be. There are many well-known problems which are encountered when using HTML for storing data on servers. Still, HTML, with its platform independence, is excellent for the exchange of data between server and client. We believe that the HM-Data Model could provide a new and useful paradigm which would eliminate the problems caused by the fact that HTML documents have links embedded in them. Using each model for that for which it is best suited seems like an ideal solution to many of the problems of the Web today. This idea could be

accomplished by mapping the rich structure of the HM data model onto the simple node-link paradigm found in HTML.

HM-Card, a software package that implements this model, as well as help files and a number of courseware examples are available from: <http://www.iicm.edu/hmcard>

7. References

[Berk and Devlin 1991] Berk E., Devlin J.(1991) Hypertext/Hypermedia Handbook. McGraw-Hill, New York.

[Goodman 1990] Goodman D. (1990) The complete HyperCard 2.0 Handbook. Bantam Computer Books, New York.

[Hall 1995] Hall T. L. (1995) Utilizing Multimedia Toolbook 3.0 Boyd Fraser Publishing, New-York.

[Maurer and Scherbakov 1996] Maurer H. and Scherbakov N. (1996) Multimedia Authoring for Presentation and Education: The Official Guide to HM-Card. Addison-Wesley Publ.Co. Bonn.

[Maurer et al. 1994a] Maurer H., Philpott A. and Scherbakov N. (1994) Hypermedia systems without links. Journal of MCA 17 (4), 321-332.

[Maurer et al. 1994b] H. Maurer, N. Scherbakov, K. Andrews (1994) Object-Oriented Modeling of Hyperstructure: Overcoming the static link deficiency. Information and CoursewareTechnology 36 (6), 315 - 322.

[Maurer et al. 1995] Maurer H., Scherbakov N. and Schneider A. (1995) HM-Card - a new Hypermedia Authoring system. Journal for Multimedia Tools and Applications, 1 (3), 305-326.

[Sims 1994] Sims R. (1994) Authorware with Style. Bantam Computer Books, New York.

[Vaughan 1994] Vaughan T. (1994) Multimedia--Making it Work. Osborne Publishing (McGraw Hill), New York.