

Introducing Hypermedia Composites to WWW

Denis Helic, Hermann Maurer, Nick Scherbakov

Institute for Information Processing and Computer Supported New Media (IICM), University of Technology Graz, Austria

This paper analyses current trends in hypermedia data modelling and suggests that composites are a necessary ingredient for any efficient authoring paradigm. The paper describes the motivations for using composites on an authoring level. It points out that WWW defines the notion of nodes and links, but it does not support composite components. It also examines the interplay between standard navigation and composite navigation. The paper offers a possible strategy for practical use of Hypermedia composites on the WWW.

1. Introduction

In an informal sense, a database is a set of data we create and maintain for reference and analysis. In a technical sense, a database imposes certain well-defined structures and operations on a data set so that creating, finding, analysing, updating and maintaining the consistency of data is made as easy and as efficient as possible, particularly for very large data sets. In both senses, hypermedia systems are database applications. [1, 2]

In its most general sense, a *Logical Data Model* defines application independent (i.e. generic) data structures and operations which can be applied to basic data constructs. We can also define data model as a user's perception of a functionality of information system as such. [1]

For example, the well known Relational Data Model defines such functionality of a data base system as a set of operations (Relational Algebra) or Rules of Inference (Relational Calculus) which can be applied to a predefined number of two-dimensional tables usually called relations. [1]

When discussing Hypermedia Data Modelling such well-known hypermedia system as the World Wide Web (WWW) must be taken into account. WWW is organised as a very big collection of so-called HyperText Mark-Up Language (HTML) documents, accessible via the HyperText Transfer Protocol (HTTP) and addressable by Universal Resource Locators (URL). Links in WWW are embedded into HTML documents as special TAGs containing an URL of a document which is referred to. [3, 4, 5]

We should distinguish two aspects of this system [1, 6, 7, 8, 9, 10]:

- data protocols which provide compatibility of different software components participating in the global WWW network (i.e. different clients and servers);
- data models which provide particular logical views of the global WWW database, i.e. reflect the user's perception.

In fact, any system which accepts HTTP requests containing an URL as a particular document identity, and which replies with an HTML document, is fully WWW compatible. As a logical data modelling paradigm, WWW supports the so-called node-link logical data model, i.e. the basic structuring of data into nodes and links, and the operation of link navigation known as browsing. [1, 7, 5]

Thus, a particular WWW server is not obliged to support the node-link paradigm internally. It can, for example, utilize a Relational Data Model or some other data model to store data. In this case, such "advanced" WWW servers just map dynamically any incoming HTTP request into internal operations and, of course, present resultant data as HTML documents. The mapping mechanism is implemented in a so-called *rendering engine* running as a front-end application on a remote server. [6, 3, 11]

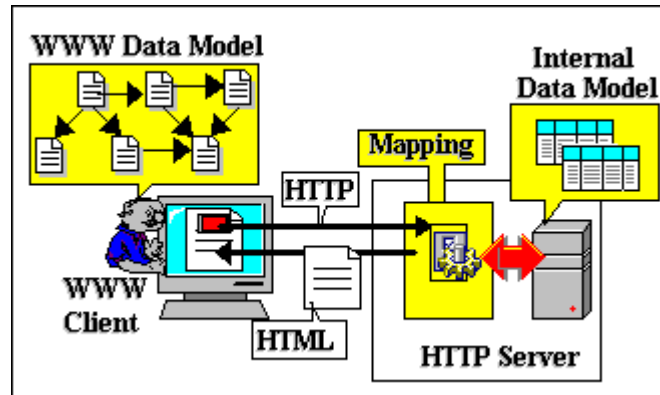


Figure 1: Mapping Internal Data Model onto WWW model

This approach has been successfully implemented in a number of commercial products (most notably, Home [11], Oracle [12] and Hyperwave [13, 14]).

Unfortunately, the HTTP protocol is very closely coupled with the Node-Link model and does not facilitate development of WWW servers based on more advanced data structuring paradigms. Thus, on the one hand, Hypermedia Data Modelling is increasingly important for reducing cost and improving quality of hypermedia design and development. Hence the practical evaluation of different data modelling concepts is a necessity. On the other hand, development of such advanced WWW servers is a multi-million undertaking which cannot be carried out just for scientific purposes.

In this context, we offer to distinguish between two components of logical data models - data updating and data retrieval facilities. Data updating facilities can be seen as special authoring paradigms and, thus, can be implemented as stand-alone software packages. A resultant database implemented by means of such authoring packages and uploaded into an HTTP server, can dynamically be mapped onto the node-link data structure and presented to a client.

The remainder of this paper consists of three main sections. The first considers hypermedia authoring and current trends in hypermedia data modelling. In this section, we present the notion of so-called Hypermedia Composites for further consideration as a main authoring concept. The second chapter describes a particular logical data model based on a special type of hypermedia composites - structured collections. The remaining section describes a working implementation of the model.

2. Current trends in Hypermedia Data Modeling

The basic node-link paradigm which is utilized by WWW, is based on two main entities - nodes and links. Nodes are multimedia documents where both structure and content is defined by means of the HyperText Mark-up Language (HTML) that was specifically designed for WWW. This is the first crucial point for our analysis: WWW does not separate a document's structure from its content. In WWW, documents are devoid of any attributes separated from an actual content data as is usual for information objects in conventional database systems: this is a second point which we would like to stress. Links in WWW are defined between so-called anchors (source and destination anchor, respectively). The source anchor is embedded within the content data. Thus, such links are bound to a particular destination anchor by some form of mark up within the content data. Links are also devoid of any attributes.

If we compare all existing proposals for new hypermedia data modelling paradigms with the previously discussed node-link model, we see the following main trends:

- (i) Extending the basic hypermedia thesaurus containing notions of nodes, links and anchors with new data structures - collections, structured collections, composite components, compounds, containers or whatever. Such new data structures are addressable entities containing other data structures, nodes and links as elements. [15, 16, 17, 18, 19] We will call such data structures *composites* in the further discussion.
- (ii) Providing some metastructuring mechanism (often, referred to as *templates*) to predefine the structure of a number of multimedia documents, thus separating the structure from the content data. [20]
- (iii) Providing multimedia documents with some attributes in addition to the content data ([Maurer 98], [Lennon 97], [Andrews 95]).
- (iv) Upgrading links to independent, addressable objects separated from document content ([Maurer 98], [Lennon 97], [Maurer 96a]).
- (v) Extending the notion of an anchor by associating procedures which locate documents at run-time and even dynamically generate destination documents at run-time [Maurer 96a].

3. HM Data Model

The HM-Data Model is a logical Hypermedia Data Model. Hence, it is less concerned with the internals of a page than it is with data structures and operations which can be applied to create, modify and explore (i.e. render) such data structures. In the following, therefore, we will consider multimedia pages as atomic, i.e. as basic indivisible chunks of multimedia data that have to be organised. We will examine the use of a novel method of hyperlinking for structuring such information chunks into hypermedia databases.

3.1 Data Structures

A database, according to the HM-Data Model, consists of addressable composites called Structured Collections (S-collections or just collections, for short). An S-collection does nothing but encapsulates members together with some internal structure (i.e. navigational topology). This structure is in fact a link structure expressing the relationships or associations between members. A member is either a page or another S-collection. One member in an S-collection is chosen and designated by the author as its *head*.

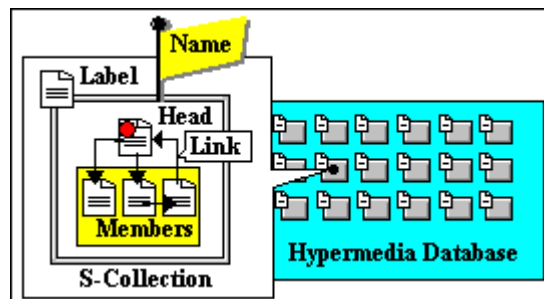


Figure 2: Internal Structure of an S-Collection

Additionally, an S-collection may have an associated page called its label (see Figure 2) to provide contents synopsis. If a label is missing then the head is used instead.

3.2 Browsing

We may think of an S-collection as an opaque container: if we are outside the container, its members will not be visible to us. To see what's inside it, we must enter it. Of course, we can only be inside one container - the current container - at any given time. But once inside, we will be able to visit its members by navigating its link topology (see Figure 3).

A member we visit in the current container may be a page or another S-collection.

- If a page, it will be visualised in some appropriate way - typically involving the presentation of the page's media objects on the computer display and/or sound system.
- If an S-collection, its label (which is a page, by definition) will be visualised. If a label is missing then the head is used instead.

The most recently visited member of the current container is the current member. Access to other members from the current member is determined by the container's link structure: only links emanating from the current member can be selected. All such links are typically visualised as anchors on the current display (these may be icons, hot-words, push-buttons, etc). Navigation within the current container therefore is from member to member as allowed by its link topology.

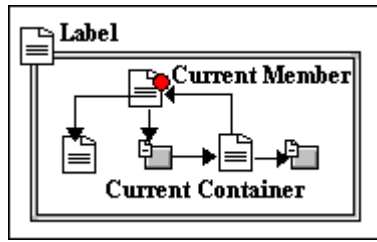


Figure 3: Browsing a Current Container

Note that visiting (i.e. navigating to) an S-collection does not enter it, even though its label is visualised. The point of the label, as mentioned earlier, is to present a synopsis of the collection to allow us to decide whether or not we want to enter it.

To enter it, we must explicitly use the Zoom-In operation. Zooming into an S-collection makes it the new current container, its head the new current member; all links emanating from the current member become accessible.

Figure 4 illustrates the Zoom-In operation. Prior to zooming in, "A" is the current container and "B" the current member. Applying the zoom-in operation on "B" makes it the current container, its head the current member, and all the links emanating from the head accessible. Navigation after zooming in is restricted to the links in the current container, or zooming into the current member (if it is a collection) or zooming out of the current collection.

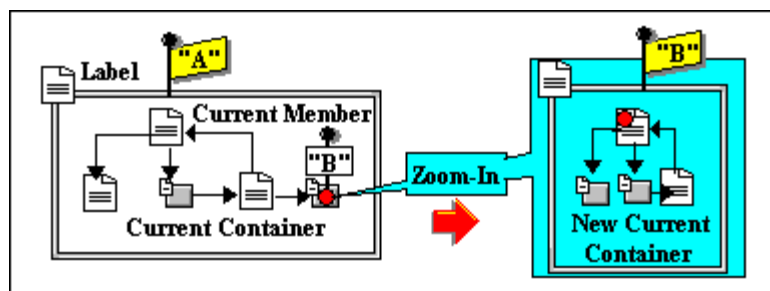


Figure 4: The Zoom-In Operation

The Zoom-Out operation is of course the inverse of Zoom-In. Its effect is to restore the current container and the current member to the state immediately prior to the most recent zoom-in. Thus, a zoom-out operation for the situation in Figure 4 will reinstate "A" as the current container and "B" as the current member.

Navigation may thus be viewed as occurring in and between "planes". Link navigation within an S-collection is in one plane, while the zoom-in and zoom-out operations are orthogonal to this and effect a jump to a lower or higher plane.

Note also that an S-collection may be a member of more than one other S-collection, i.e. collections may be re-used in different contexts. A zoom-out operation, however, will return to the plane from which it was reached. Thus, if an S-collection "X" (see Figure 5) was zoomed in from "B", zooming out gets back to "B" and not "A". This preserves therefore the context of viewing an S-collection. Figure 5 illustrates such Navigation "Planes".

One other orthogonal navigation operation is defined in the model: the Zoom-Up operation. This operation effects a jump from the current container to another that also contains the current member. As there may be more than one such container, the operation must also specify which.

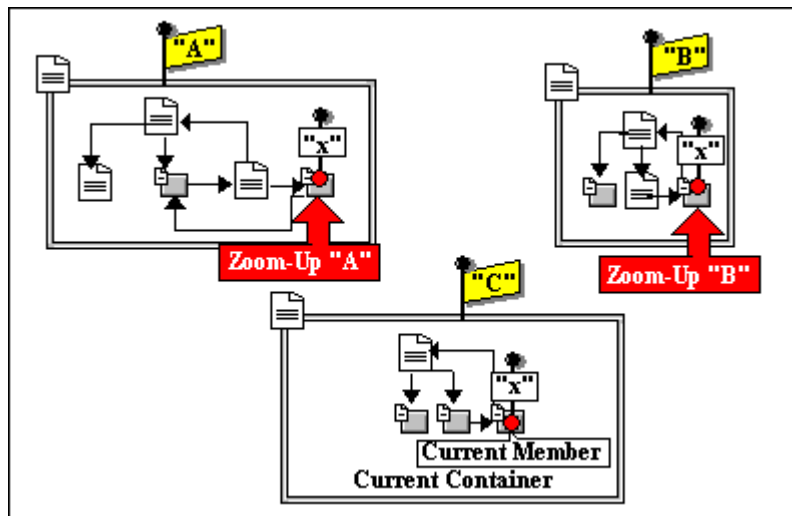


Figure 5: Navigational Planes

3.3 Data Classes

In the HM-Data Model, a particular S-collection is an instance of one of four predefined types of S-collections:

- (a) Folder,
- (b) Envelope,
- (c) Menu, and
- (d) Freelinks

Each type or data class defines a particular link topology which constrains the way members can be linked to one another. Additionally, the integrity of the links in any of these types will be automatically maintained.

Figure 6 illustrates the link topology of each type:

- the Folder connects each member in a bidirectional circular list
- the Envelope links each member to every other member

- the Menu connects the collection head to every other member using bidirectional links
- the Freelinks allows members to be linked in any way desired by the author; the author must ensure in this case that there is a path from the collection head to each member

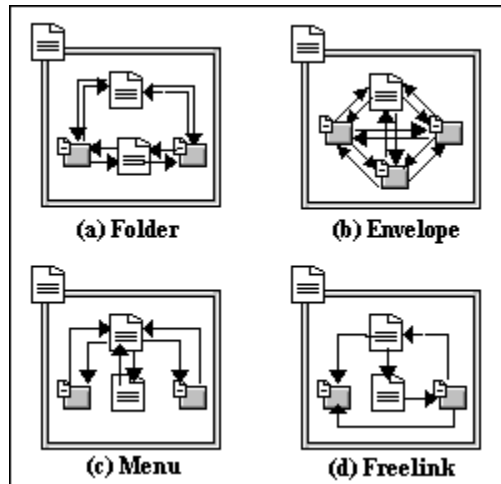


Figure 6: Classes of S-Collections

3.4 Creating and Modifying a Database

Building a hypermedia database typically follows a bottom-up approach. First, multimedia pages are created. They then become building blocks for complex S-collections that constitute the database.

Assuming that pages have been created, an author creates a new S-collection by selecting the desired type (i.e. a folder, envelope, menu or freelinks), assigning it a unique name, selecting a page or an existing S-collection as its head, and optionally selecting a page as its label.

Once an S-collection has been created, members can be inserted, modified or removed. These operations will automatically update the link structure according to the chosen collection type. Insertion of a new member into a menu collection, for example, will automatically create a bidirectional link between it and the collection head. Conversely, removing a member will remove its corresponding link to/from the head. Similar automatic creation and deletion of links apply to envelopes and folders. With folders, of course, we must also specify the insertion position in the circular list. The exception to all this is the freelinks collection, where the user must explicitly insert and/or remove links to create the desired link topology. Links are second-generation, i.e. separated from rather than embedded in the members. This is of course essential since a member can be re-used in another collection with a different link topology.

4. Introducing the HM-Data Model to WWW

Formally, a particular type of an S-Collection (i.e. Folder, Menu, etc.) is a meta definition of a specific linking structure which is automatically supported by all instances (i.e. S-Collections) of this type. Thus we can say that an S-Collection automatically imposes a particular navigable structure on a top of collections of existing HTML pages or other S-Collections defined as members.

Practically speaking, we can understand an S-Collection of any type as an entity comprising three elements: label, head and a set of members (see Fig. 2). Such a data structure may be simply visualized as a special template consisting of three cells. Similarly, an instance of such data type (i.e. S-Collection) might be seen as a template filled with existing HTML documents and/or other existing S-Collections (see Fig. 7).

Thus, from an author's point of view, there is a number of predefined templates (S-Collection types) where the author can simply insert existing pages or other S-Collections to define sophisticated navigable structures.

Informally speaking, authors just select previously created pages and/or S-collections from their local file system drag and drop them into other containers (S-collections); links are generated automatically.

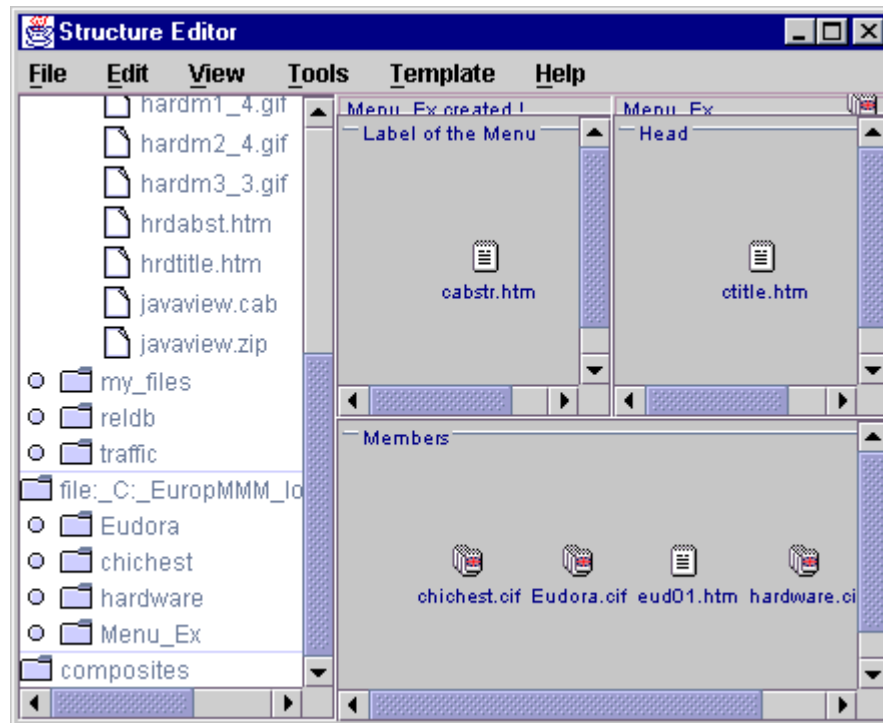


Figure 7: Authoring an S-Collection

As stated earlier the Internet Browsers (like Netscape and MS Explorer) of users do not access directly S-Collections. Instead, in order to obtain data, the browser communicates with a special Rendering Engine which is a Server Side Script (if the S-Collection resides on a Hyperwave server) or a special Java Applet (if the S-Collection resides on a local drive or an ordinary WWW server).

In other words, whenever a user accesses such an S-Collection with an ordinary WEB browser a special software component is run to visualize the unit in a form of interrelated HTML pages in accordance with the algorithm presented in Section 3.2. A so-called CIF file containing a description of the generic link structure and attributes attached to the S-Collection is essentially used to control such visualization. Figure 8 shows a possible visualization of the menu authored as shown above (see Fig. 7).

Links may be visualised in a variety of ways and users, when adding or modifying a member, can typically select how a source anchor is visualised. These may be hot-spots, hot-words, scrolling menu lists, icons (as chosen for the example on Fig. 7), etc., depending on the particular application of the model.

Of course, similar to import/export facilities of many database systems, subsets (of pages and/or collections) of a database may be preprocessed to generate all necessary links as HTML tags, and separately saved (the option is called "snap" in this particular application). Such a "hardwired" S-Collection can be accessed directly by an internet browser without any additional rendering engine and can hence be used separately. It should be clear, however, that copies are actually made in these cases

and that the separated hyperstructure is maintained independently, i.e. changes made in source pages and/or S-Collections cannot be seen in this "hardwired" hyperstructure.

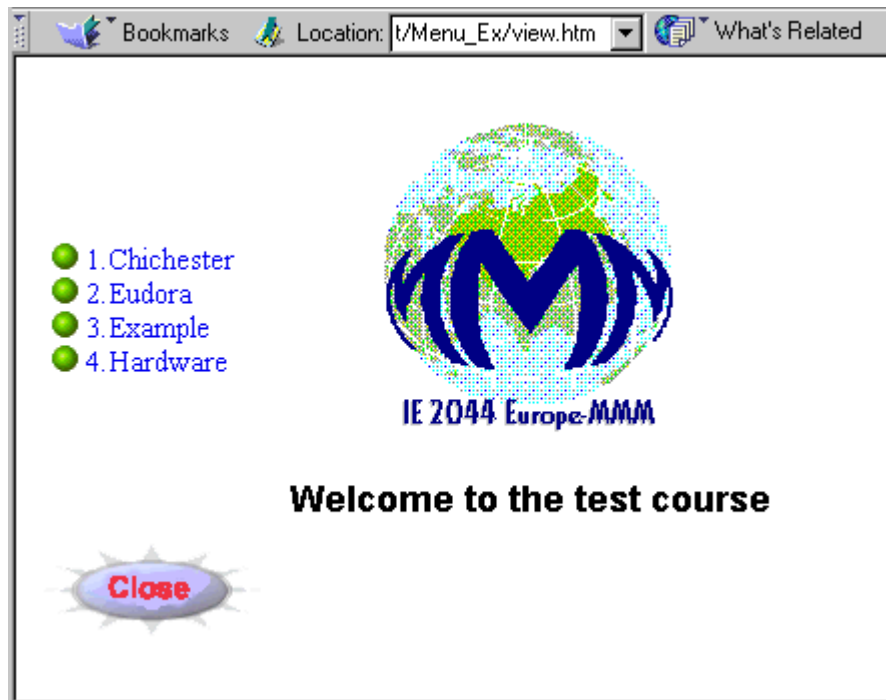


Figure 8(a): Rendering an S-Collection (The menu becomes a current container)

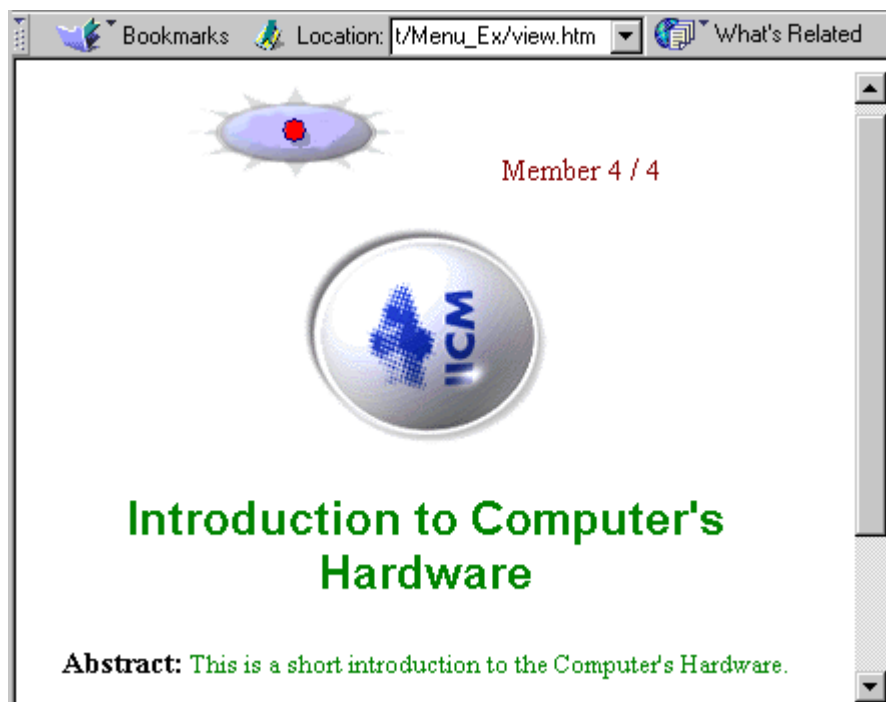


Figure 8(b): Rendering an S-Collection (The member "Hardware" is accessed)

5. Conclusion

It should be apparent that the HM-Data Model is highly modular. Pages and collections can be created independently but at the same time can be re-used through a flexible nesting or containment mechanism. This greatly facilitates multi-author development of hypermedia databases, involving mainly:

- creating pages (using some appropriate HTML editor application)
- creating S-collections of appropriate type (determining therefore their internal link structure)
- re-using previously created pages and S-collections to define the contents of S-collections being created
- storing new S-collections into a remote server or local file system (thus making them available for re-use in other S-collections)

Any S-collection can be modified at any time and, as mentioned above, internal link associations will be automatically updated as members are added/removed. Note, however, that removing a member does not actually delete it from the database. To delete an S-collection, we must do so explicitly. All collections containing the one being deleted will be updated accordingly.

The HM-Data Model essentially replaces the "spaghetti" view of the basic Node-Link model with more structured, independent but fully compatible hypermedia modules called S-Collections.

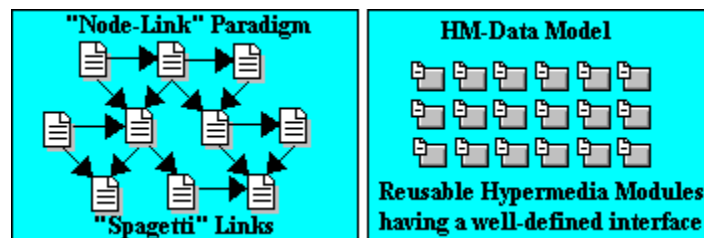


Figure 9: HM-Data Model vs. Ordinary Node-Link Paradigm

In summary, the following features of the HM-Data Model distinguish it from other existing hypermedia data models:

- Links neither belong to individual nodes nor are they globally addressable objects. Instead, they are encapsulated in hypermedia containers called S-collections. By definition, links exist only between members of a collection, i.e. links cannot be created to destination nodes that are outside the collection. S-collections therefore represent well-defined chunks of information that may be re-used in different contexts without concern for superfluous links.
- The restriction of links to local contexts only is compensated by orthogonal navigation operations of Zoom-In, Zoom-Out and Zoom-Up.
- Containment of complex collection objects within another, referred to as "re-use" rather than "reference", is in line with object-oriented views and closer to the intended concept of sharing existing resources, particular in different contexts. The term "reference" too strongly suggests "jumping to another location" (with a fixed given context). "Re-use" on the other hand implies a (logical) embedding of the external object into the current context (switching between embedding contexts is via Zoom-Up).
- Authoring is effected by memberwise inclusion of hypermedia chunks, as opposed to "spaghetti" linking of nodes in other models. Any S-collection can be included in any other. Recursive membership relations remain a possibility and will allow the modelling of arbitrarily complex hypermedia databases.

- All operations are addressed to a particular data object (S-collection) and do not affect the link structure of other objects. This object-oriented character of the model presents new ways of supporting the logical integrity of hypermedia databases.

References

- [1] H. Maurer, N. Scherbakov: *From Databases to Hypermedia*. Springer Publ.Co. Berlin. 1998.
- [2] J. Nielsen: *Hypertext and Hypermedia*. Academic Press, 1990.
- [3] K. Andrews, A. Nedoumov, and N. Scherbakov: *Embedding Courseware Into Internet: Problems and Solutions*. Proceedings of ED- MEDIA'95, Graz, Austria, June 1995, pp.69-74.
- [4] R. Cailliau: *About WWW*. Journal of Universal Computer Science, 1:221-230, Apr 1995.
- [5] T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, and A. Secret: *The world-wide web*. Communications of the ACM, 37:76-82, Aug 1994.
- [6] M. Fernandez, D. Florescu, J. Kang, A. Levy and D. Suci: *Strudel: A Web-site management system*. Proceedings of ASM SIGMOD'97 Tucson, Arizona, May 1997 pp. 549-552.
- [7] J. Lennon J.: *Hypermedia Systems and Applications - World Wide Web and Beyond*. Springer Verlag, Berlin. 1997.
- [8] G. Bucci, R. Detti, V. Pasqui, and S. Nativi: *Sharing multimedia data over a client-server network*. IEEE Multimedia, 1(3):44-55, 1994.
- [9] H. Eichmann, D. McGregor, and J. Danley: *Integrating structured databases into the web: The more system.*: Proceedings of the First International WWW Conference. Geneva, Switzerland, May 1994, pp. 369-378.
- [10] S. Khoshafian, A. Chan, A. Wong, and H. K. T. Wong: *Client-Server SQL Applications*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 1993.
- [11] E. Duval, H. Olivie, P. Hanlon, D. Jameson: *HOME: An environment for hypermedia objects*; Journal of Universal Computer Science, 1:154-173, May 1995.
- [12] M. Gwyer: *Oracle Designer/2000 Webserver Generator Technical Overview (version 1.3.2)*. Technical Report, Oracle Corporation, Sept. 1996.
- [13] H. Maurer (Ed.): *HyperWave: The Next Generation Web Solution*. Addison-Wesley Longman, London. 1996.
- [14] F. Kappe, H. Maurer, and N. Scherbakov. *Hyper-G. a universal hypermedia system*. Journal of Educational Multimedia and Hypermedia, 2:39-66, 1993.
- [15] H. Maurer, N. Scherbakov: *Multimedia Authoring for Presentation and Education: The Official Guide to HM-Card*. Addison-Wesley Publ.Co. Bonn. 1996.
- [16] H. Maurer, N. Scherbakov, K. Andrews, and P. Srinivasan: *Object- Oriented modelling of hyperstructure: Overcoming the static link deficiency*. Information and Software Technology, 36:315-322, 1994.
- [17] L. Hardman, D. C. A. Bulterman, and G. van Rossum: *The amsterdam hypermedia model - adding time and context to the Dexter model*. Communications of the ACM, 37:50-62, Feb 1994.
- [18] N. Streitz, J. Haake, J. Hanneman, A. Lemke, W. Schuler, H. Schutt, M. Thuring: *SEPIA: A Cooperative Authroing Environment*. Proceedings ACM ECHT'92, Milan, Italy, Dec. 1992 pp 11-22.

[19] F. Garzotto, P. Paolini, D. Schwabe: *HDM - A Model for the Design of Hypertext Application*. Proceedings ACM Hypertext'91, St. Antonio, Texas, Dec. 1991. pp.47-58

[20] P. T. Zellweger: *Scripted Documents: A Hypermedia Path Mechanism*. Proceedings. ACM Hypertext '89, Pittsburgh, PN, Nov. 1989 pp 1-14.