Computer Science and Systems Analysis

Computer Science and Systems Analysis

Technical Reports

Miami University *Year* 1992

Node Re-Usability in Structured Hypertext Systems

Omer Abdalla* Fazli Can†

*Miami University, commons-admin@lib.muohio.edu

†Miami University, commons-admin@lib.muohio.edu

MIAMI UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & SYSTEMS ANALYSIS

TECHNICAL REPORT: MU-SEAS-CSA-1992-006

Node Re-Usability in Structured Hypertext Systems

Omer Abdalla and Fazli Can

School of Engineering & Applied Science | Oxford, Ohio 45056 | 513-529-5928

Node Re-Usability in Structured Hypertext Systems

by

Omer Abdalla Fazli Can
Systems Analysis Department
Miami University
Oxford, Ohio 45056

Working Paper #92-006 August, 1992

NODE RE-USABILITY IN STRUCTURED HYPERTEXT SYSTEMS

Omer ABDALLA **Fazli CAN***

Department of Systems Analysis
Miami University
Oxford, OH 45056

August 2, 1992

Abstract

When the size of a graph-based hyperdocument exceeds a certain limit, the graph structure gets complicated and causes navigation and document management problems. A simple solution for this problem is the structuring of the hyperdocument into several smaller units. In this approach each unit contains nodes that share common properties and their link structures. Smaller, more manageable networks (called webs) which have their own, less complex graph structures are the result. In this paper we propose a model for hypertext systems which allows hyperdocument structuring using webs. The model demonstrates node re-usability which becomes essential as it is very likely that the smaller units created will share nodes. The implementation details of a hypertext authoring system, HypAS, based on the proposed model is also provided.

1. INTRODUCTION

The term "hypertext" is coined by Ted Nelson [NELS67] to refer to systems that would permit users to track down information following non-linear paths through text. Hypertext systems provide a non-sequential interactive mechanism for navigating through textual documents (or non-textual documents in hypermedia systems) [NIEL90]. This mechanism is meant to provide a more effective way of presenting information to the user.

The hypertext concept was introduced in 1945 when Vannevar Bush [BUSH45] proposed his "memex" machine in an article titled "As we may think." Bush's idea inspired the implementation of a computer-based hypertext system called "NLS," in 1968 by Doug Engelbert, another pioneer in the hypertext field [ENGE68].

Hypertext systems are modeled in many ways. Three models are found to be useful in

* (513) 529-5950, fc74sanf@miamiu.bitnet

three distinct areas of implementation: the Dexter model, the gIBIS model, and the Trellis model. The Dexter model is presented as a model of hypertext structure [FRIS92]. It is composed of three separate layers: a storage layer which defines how text components and links can be related to form hypertext networks; a within-component layer which provides a mechanism to define the composition of each type of component; and a runtime layer which provides a mechanism to define how hypertext components and links will be displayed and manipulated. The gIBIS model (generalized Issue-Based Information System) is a hypertext implementation based on IBIS - a rhetorical model that allows one to relate issues, arguments, and positions by means of predefined set of components and semantic links [CONK89]. The Trellis model is a generalization of existing directed graph-based forms of hypertext [STOT89]. It is based on the Petri net representation of a document structure. The model also specifies the browsing semantics to be associated with the hypertext.

In this paper we present a new model based on the concept of node re-usability. Section 2 presents the Node Re-usability Model. It describes the four layers of the model: the Presentation layer, the Node layer, the Link layer, and the Storage layer. It also demonstrates hyperdocument network structuring and shows how it benefits from the Node Re-usability model. Features of the Node Re-usability model and its advantages are also presented in this section. Section 3 presents HypAS, a prototype implementation of the Node Re-usability model. The four subsystems of HypAS: the user interface subsystem, the nodes subsystem, the links subsystem, and the file and storage subsystem are described in detail. HypAS node and link structures and their advantages are also presented. Section 4 summarizes and concludes the paper

2. THE NODE RE-USABILITY MODEL

A typical hypertext system consists of nodes, which may contain any visual or audio information and a way of storing and retrieving nodes and link structures. Nodes are linked with each other in a network structure. The user can traverse through the network and jump from one node to another in a non-sequential manner using links. The collection of nodes and their

link structures constitute a hyperdocument. A hyperdocument may consist of either a single- or multi-network graph. In a single-network hyperdocument, all nodes and their link structures create a single graph. In multi-network hyperdocuments, however, nodes and link structures are made of more than one graph structure.

2.1 Layers of the Model

Our model is composed of four layers: The Presentation layer, the Node layer, the Link layer, and the Storage layer (see Figure 1).

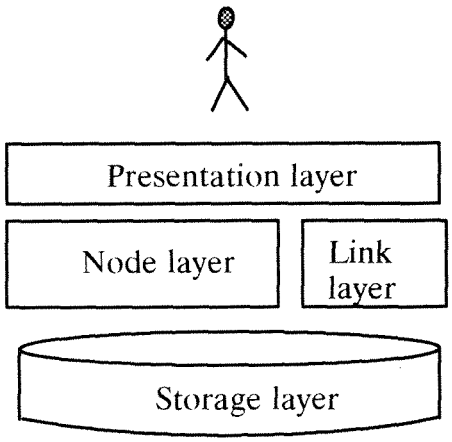

Figure 1. The four layers of the Node Re-usability model.

The Presentation layer is a high level layer of abstraction built on top of the other three layers. It hides implementation details from the user, and provides a user-friendly interface. Browsing strategies such as single-web or multi-web browsing are implemented at this level. The use of a mouse, a keyboard, or any other interactive media is also specified at this level.

The Storage layer is at the bottom of the model structure. It is a low-level detailed description of the system components (i.e nodes and link structures). The model does not suggest any strategy for implementing this layer. The only requirement is the separation between link structures and node storage.

The Node and Link layers are placed between the Presentation and Storage layers. The Node layer contains definitions and specifications of the node types supported by the system as well as the methods and operations for presenting the node to the user. For example, a

command node (command nodes are explained in section 3.2.4) requires the operating system command interpreter to handle it. We should specify the interface between the hyperdocument and the command interpreter at this level. Nodes are considered as independent, completely re-usable objects in this model.

The Link layer contains definitions of links and link structures. Methods for creating and removing links or link structures are specified in this layer. Links are created according to the type and specification of nodes (defined in the Node layer). Some nodes are capable of containing link anchors (open-nodes) while others are not (closed-nodes).

2.2 Hyperdocument Network Structuring

A hyperdocument network, G_D, can be represented by a two tuple.

$G_D = <N_D, L_D>$ where,

N_D = The set of nodes in the hyperdocument (graph vertices),

L_D = The set of links in the hyperdocument (graph edges).

The graph structure can be simple or complicated depending on the size of the document N_D and the number of links that constitute L_D. For complicated graphs the network should be divided into two or more smaller webs (see Figure 2). After structuring, each web is a subset of the original hyperdocument G_D. The web graph structure, G_W, can be represented by a two tuple.

$G_W = <N_W, L_W>$ where,

N_W = The set of nodes in the web (graph vertices),

L_W = The set of links in the web (graph edges).

This can be done by cutting some of the graph edges in G_D which provide connection among webs to be created. For each web, $N_W \subset N_D$ and $L_W \subset L_D$. In systems without node re-usability (i.e when the node content and link information are stored together), the duplication of the node at the other-web side of the cutting edge is required (e.g. nodes C and D in Figure 2b). In the webs of Figure 2 $|N_{W1}| + |N_{W2}| > |N_D|$ where $|.|$ indicates the cardinality of a

set.

In order to make each web graph complete, even more than two copies of a node may be required depending on the number of webs cut at the node. All copies of the node should be modified to reflect the link structure of the specific web in which each copy belongs. Of course, this is true only for open–nodes since closed–nodes are shareable by nature without any duplication.

This network structuring approach suffers from three major problems.

(1) Node duplication is a waste of storage space.

(2) The process involves potential inconsistencies which may result from node modification.

(3) It is hard to establish connection among webs, and the sense of a single hyperdocument may be lost, since the copies of the connection nodes are considered as new nodes in some operating system environment (e.g. in MS–DOS, file versioning is not supported).

The Node Re–usability model is immune to any of these problems. First, there is no additional storage cost in terms of nodes as a result of network structuring because there is no need for the node duplication process. Open– and closed–nodes are all shareable among all webs without the need of duplication. Hence, $|N_{W1}| + |N_{W2}| + \ldots = |N_D|$. Second, the inconsistency problem which may result from the use of multiple versions (copies) of a node is eliminated in this model for the same reason above. Third, the connection among webs is straightforward and it can be established at any point where a shareable node is reached. This is possible, because web link structures are loaded into memory prior to establishing the connection, and it is easy to conduct a memory search to find the same node in one or more of these structures.

2.3 Features of the Node Re–usability Model

The proposed model works for frame–based single–user systems, but can be modified and adapted to other hypertext environments. A hypertext authoring system based on the proposed model should exhibit the following features:

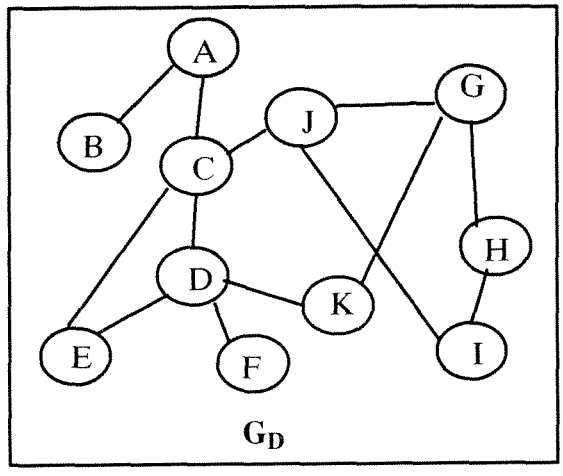

Figure 2a. A complex graph of a hyperdocument.

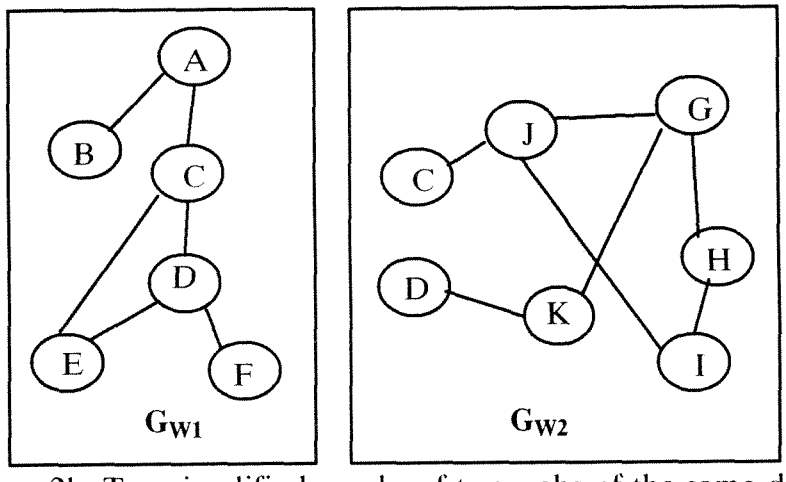

Figure 2b. Two simplified graphs of two webs of the same document (Note that the hyperdocument graph is cut at node C and D).

Figure 2. An example for the construction of webs from hyperdocuments.

(1) Bottom-up Hyperdocument Construction. The hypertext author should be able to construct hyperdocuments in a bottom-up manner starting with node creation, then linking each group of nodes in a separate web, and finally constructing the hyperdocument by calling these webs from within one another (in the presentation level).

(2) Link-Node Separation. Link information should be separated from the node content. Link information may include the anchor text and position, destination node, etc; and links should be added or removed without modifying the node.

(3) Node Re-usability. Each node should be made re-usable in more than one web without any duplication.

(4) Multi-web Browsing. The system should provide a mechanism for loading and traversing through more than one web at a time. It should also provide the ability of

switching from one web to to another without difficulty.

This may seem to be a memory-bound approach especially if both node and web link structures are to be loaded into the main memory simultaneously. However, in implementation each node can be saved as a separate file and loaded only when needed. Only web link structures are kept in memory until they are unloaded by the user.

2.4 Advantages of the Node Re-usability Model

The major advantage of the model is re-usability which saves storage space and development time. Other advantages are the following.

(1) The hyperdocument maintenance is now more easier because of the link-node separation. Nodes can be added, modified, or deleted at any time. Simple integrity constraints can be enforced to prevent deletion of shareable nodes.

(2) Complete webs as well as nodes may be ported to other hyperdocuments with ease.

(3) Multi-web browsing eliminates the navigational problems of single-graph large systems because switching from a web to another gives the feeling of a physical jump from one place to another in the hyperspace.

3. HypAS A PROTOTYPE OF THE NODE RE-USABILITY MODEL

HypAS (Hypertext Authoring System) is a prototype implementation of the Node Re-usability Model described in the previous section. It is a frame-based hypertext authoring system that works on any PC running MS-DOS 3.0 or later versions. HypAS is implemented in the C language. It consists of four major components:

(1) User interface subsystem,

(2) Nodes subsystem.

(3) Links subsystem,

(4) File and storage subsystem

In this section each component is described in some detail.

3.1 User Interface Subsystem

Two distinct user interfaces are available in HypAS. One for the hypertext reader and the other for the hypertext writer (or author). Both interfaces are designed under one common phi-

losophy: ease of use. They both support the use of the keyboard and mouse for invoking HypAS commands.

3.1.1 The Hypertext Reader Interface

This interface corresponds to the Presentation layer of the Node Re-usability model. In this interface the user is provided with all available navigational tools. The arrow movement keys are used to jump from a highlighted button to another and the < Enter > key is used to invoke the node pointed to by the highlighted button. Alternatively, the mouse may be used to do the same task.

Buttons are displayed in colors that differ from the rest of the text. For each type of node the reader may select a different highlighting color for the button that points to it because the colors can be re-configured. No typing is required or allowed via this interface except for keyword searching.

The user interface, when first called, provides the reader with a list of web files to select from, since a hyperdocument may consist of more than one web. The reader can select one or more webs for loading. When a web is selected, its network link structure is loaded into the main memory. The first screen of the first web loaded is displayed and the link buttons (anchors) are highlighted. Other screens are displayed according to the selection of the user. The user may return to the main menu at any point to load or unload additional web files. If a new web is loaded and it contains the last node accessed in the previous web, then that last node becomes the current node of the new web (for synchronization purposes). The new web becomes the driver of the hyperdocument browsing. Otherwise, the driver is the previous web. The user may activate any previously loaded web and make it the driver. It is also possible to specify all webs of a particular hyperdocument in a hyperdocument file. If the hyperdocument file is selected then all its webs are loaded and the control is given to the first web specified.

In a particular web, the reader may go sequentially through the screen nodes in the forward or backward direction using < PgDn > and < PgUp > keys. He/she may jump to the first node using the < Home > key or to the last node using the < End > key. The reader may

also back-track up to the last 100 screen nodes visited using the <Esc> key. Of course the reader may select highlighted buttons (anchors) to traverse to nodes not in the sequence outlined above.

The reader may invoke a menu system that provides extra commands such as configuration of button colors, loading or unloading other webs, searching the web network for specified keywords, going to the DOS operating system or quitting the hypertext reader.

3.1.2 The Hypertext Author Interface

This interface provides a workspace (or window) to be used for creating and editing screen and index nodes, browsing and running network webs, creating and removing links, and editing and running script files. The workspace occupies the first 23 lines of the screen. A user-friendly menu system provides all authoring commands in the bottom two lines of the screen. Four authoring modes are supported by HypAS: The Painter mode, the Linker mode, the Web mode, and the Script mode. The default authoring mode is the Painter mode. The system switches automatically to the other modes when necessary. Although the Hypertext Author Interface corresponds to the Presentation layer of the Node Re-usability model (in the Web mode), it is also the interface for the author to access the other three layers: the Node, the Link, and the Storage layers. It provides access to the subsystems that corresponds to these three layers, namely, the Node subsystem, the Link subsystem, and the File and Storage subsystem

While in the Painter mode, HypAS provides a screen editor which is used for creating and editing screen and index nodes. The editor is equipped with a color palette and an ASCII chart which pops up for the user to select from when the assigned function key or menu command is called. It also has line and box drawing facilities. Block operations (i.e copy, move, erase, color, and outline) are also supported. The editor is capable of importing and exporting small ASCII files to and from the screen and index nodes.

In Linker mode, the author may create or remove links from the screen or index nodes. He can view the link information and modify it. When this mode is active, link anchors are

highlighted.

The Web mode provides the user with the necessary tools to load, view, edit and test a particular web link structure.

The Script mode enables the user to load, edit and run script files using the HypAS workspace.

3.2 HypAS Nodes Subsystem

In this subsystem all nodes supported by HypAS are defined. Methods to access and present them to the users are also defined internally. HypAS supports a variety of nodes some of them are created by the built-in editor. With the exception of note nodes, all nodes are considered to be independent from each other and from the hyperdocument in HypAS. They are created and stored in separate files with no association to any link information. Nodes are created or made available prior to linking them in any web. All nodes except screen and index nodes are reusable by their nature because no link anchors can be embedded in them (i.e. they are closed-nodes). The following is a list and explanations of node types available in HypAS.

Screen Nodes

Screen nodes represent the basic building blocks of HypAS hyperdocuments because they support the creation of anchors which can link to any type of node supported by HypAS (i.e. they are open-nodes). Each node is a 23 rows by 80 columns screen which may be filled with any combination of ASCII characters (including graphic characters) and color attributes. Each node is saved in a separate MS-DOS binary file. When note nodes are created, their link information is stored at the tail of the screen node file. For re-usability purposes, no other link information is stored in screen files.

Note Nodes

Notes are small chunks of information that can be popped-up in small windows when their buttons are selected. The links to note nodes are always active in all webs that use their host screen nodes. Note nodes are used mainly for storing information which is strongly associated with their host screen nodes. The user returns to the host screen with a key stroke or a mouse

click while the pop-up note window is displayed.

Graphic Nodes

Bit-mapped graphic files in the .PCX format may be used as graphic nodes, and they are only for display purposes. These files should be prepared by other programs and they must exist before links can be established to them. After viewing the image, pressing any key or clicking a mouse returns the user to the screen or index node which hosts the button that points to the graphic file. The system checks for the existence of the appropriate graphic adapter and issues an error message if it is not available.

Command Nodes

Any MS-DOS program (.EXE or .COM) or batch file (.BAT) is qualified to be a command node. Given that enough memory is available, the command node is executed when the command button that points to it from a screen or index node is selected. This allows the user to link his hyperdocument to almost any kind of MS-DOS application e.g spreadsheet, graphic animation, calculator, sound player, etc. After the execution of the program, the user returns to the node from which the program was called.

Script Nodes

HypAS comes with a built-in simple script language interpreter. The script language consists of some BASIC language commands in addition to some special commands for displaying screen nodes in various visual effects i.e fade-in, explode, instant replace, etc.

Script files are ASCII files which can be created by the built-in editor of HypAS (Script mode) or any external text editor. The user can run and test these files before linking them to the hypertext network.

Index Nodes

Index nodes are created in the same way that screen nodes are created, by using the system painter. Index nodes can link to all types of nodes supported by HypAS including the link to other index nodes. They differ from screen nodes, since the associated link information is

saved with the node itself and not in the network web file. This feature increases the number of link buttons that can be created in an index node without requiring a huge memory space.

ASCII Nodes

Any ASCII file is qualified to be an ASCII node. There is a built-in ASCII file viewer which is called to view the ASCII file linked to from a screen or index node. Link anchors may not be created in ASCII files in this version of HypAS, but it is possible to add this feature in future versions.

3.3 HypAS Links Subsystem

Once nodes are made available, they can be linked together to form a network (web) of nodes. All tools needed for creating and removing links are defined in this subsystem. A typical hyperdocument may consist of one or more webs sharing the use of some or all of the nodes. Before applying link structures, nodes can be seen as scattered pieces of information without any relationship between them. There is no limit to the number of link structures that can be applied to the same group of nodes.

The link buttons can be created only in screen and index nodes, because they are the only open-nodes in HypAS. The link information in index nodes is stored within the index node file. This is because index nodes are special kinds of nodes and they are not meant to be re-usable by webs, since an index may refer to a node not defined in some webs. However, as far as screen nodes are concerned, no link information is stored within the screen node file except for links that point to note nodes. The link information of screen nodes is stored in a separate file, namely, the web file. This approach provides the re-usability of nodes among many hypertext webs or hyperdocuments.

A simple data structure is created to store a record of link information for each screen node that contributes to the web. It is used for initial highlighting of buttons when a screen node is displayed. Note, however, that different hypertext webs may highlight different link buttons depending on the link definitions. The data structure is also used for displaying the moving bar which is used for selecting a button and for searching keywords within the network

web.

3.4 File and Storage Subsystem

The Files and Storage subsystem deals with all file manipulation techniques and storage of nodes and link definition networks. It corresponds to the Storage layer of the Node Re-usability model. As mentioned earlier all nodes except note nodes are stored in separate files.

3.4.1 File Naming Scheme

HypAS follows the standard naming conventions for imported files and adds its own name extensions to the internally developed nodes. Table I provides a list of file extensions accepted by HypAS if the files are to be used as hypertext nodes.

Table I. File extensions of the nodes supported by HypAS

File Extension	Explanation
.SCR	Screen node file, created by HypAS.
.NDX	Index node file, created by HypAS.
.PRO	Script node file, created by HypAS or any other text editor.
.PCX	Graphic node file created by any graphic painting utility that supports this format.
.COM and .EXE	Command node files which are actually MS-DOS executable programs
.BAT	Command node file which is an MS-DOS batch file.
.TXT	ASCII node file, created by HypAS or any other text editor.
.WEB	Hypertext web file, created by HypAS for storing link definitions of one network.
.HYP	A Hyperdocument file which specifies the webs to be loaded for a particular hyperdocument. It's an ASCII file created by HypAS or any other text editor.

The next sections provide some details about the data structures used in creating index

and screen nodes, the HypAS web files, and the back-track feature.

3.4.2 Index and Screen Node Structures

Both index and screen nodes are stored as binary files using similar storage structures. The first 3680 bytes are used for storing the characters that form the index or screen node and their attributes. Note that each screen is 23 lines by 80 columns. Hence, 23*80*2, (3680), bytes are required to store both the screen characters and their color attributes.

3.4.2.1 Index Node Structure

In addition to the screen image storage (the first 3680 bytes), an index node file stores a variable number of records about link definitions. Figure 3 shows the format of these records in C language.

```
struct {
  char button_text[ANCHOR_SIZE];
  int  X, Y;
    /* position of the anchor on index screen */
  int  destination_node_type;
  char destination_node_name[NAME_SIZE];
} index_links[MAXIMUM_LINKS_PER_INDEX_NODE];
```

Figure 3. Record structure of index node links.

3.4.2.2 Screen Node Structure

The record structure for storing link definitions for the screen nodes is slightly different than the record structure used to store link definition for index nodes. In screen nodes, the link information stored is only about note nodes and there is no link to any external destination node file. The structure provides the capability of storing the target information (i.e. the content of the note node) together with the link information (see Figure 4). The number of records stored is the actual number of links used (between zero and MAXIMUM_NOTES_PER_SCREEN_NODE).

3.4.3 Web Record Structure

The web file is used to store the link structures of a single network in a hyperdocument. This

```
struct {
   char button_text[ANCHOR_SIZE];
   int  X, Y;  /* position of link button on screen node */
   char note_text[NOTE_SIZE];
  /* the information to be popped-up when a node button is
     selected */
} note_links[MAXIMUM_NOTES_PER_SCREEN_NODE];
```

Figure 4. Record structure of note links in screen nodes.

information is the actual driver of browsing (see Figure 5).

When creating a new web file, the user is asked to provide the file name of the initial node. A record is created for that node in the head of the web file. Subsequent node definitions are stored in the file in the order they are added. In addition to hypertext browsing, sequential browsing according to the physical order in which nodes are stored is also possible.

```
struct {
  char screen_node_name[NAME_SIZE];
  struct {
    char button_text[ANCHOR_SIZE];
                          /* the anchor to be highlighted */
    int  X, Y;            /* and its coordinates */
    int  destination_node_type;
    char destination_node_name[NAME_SIZE];
  } screen_node_links[MAXIMUM_LINKS_PER_NODE];
} web_record[MAXIMUM_NODES_PER_WEB];
```

Figure 5. Record structure of a web file.

3.4.4 Back-track Structure

The back-track structure allows a hypertext reader to go back to the previous nodes from the recent node to the first node (by pressing a key or clicking a mouse). For implementing the back-track feature in HypAS, an array is declared for storing the names of the last 100 screen or index nodes visited. It can store 100 names and it behaves like a queue. Note that we need such a structure for each web loaded (in a multi-network system). Hence, it can be part of the web structure in Figure 5.

3.5 Advantages of the HypAS Data Structures

The advantages of the HypAS data structures are the following.

(1) They provide the re-usability of nodes. Screen nodes are completely re-usable by various hyperdocuments and webs. Link buttons are re-definable because no link information is saved with the screen node. This saves development time and storage space.

(2) Maintenance of hyperdocuments developed under HypAS is easy because hyperdocuments can be structured into webs. Each web in turn consists of several independent nodes and link structures. Thus, adding, modifying or deleting individual nodes is always an easy task and involves no risk. The system provides all necessary integrity checking such as the existence of all nodes defined in a web, and prevents destruction of shareable nodes.

(3) Memory requirement is less than comparable systems because nodes are not loaded into memory until they are needed (nodes are loaded one at a time). The only data to be kept in memory are the network link definitions and the back-track information.

(4) Screen and index nodes are very small files and loading them into the computer memory is almost instantaneous (the user would not feel any I/O operations or delay in the system's response).

(5) Both sequential and non-sequential hypertext browsing is supported.

The re-usability of nodes using the one-file-per-node approach limits the size of hyperdocuments by the maximum number of files allowed in an MS-DOS directory. However, this does not cause any practical problem since different file types are stored in different directories (the HypAS installation utility creates these directories). There is a separate directory for each node type. The author also has the option of storing all files in one directory if his hyperdocument fits in.

A field showing the folder or directory which contains the specific node may be added to the web record structure. This would expand the addressability span of the web links and makes it possible to invoke nodes from any place in the storage space.

The implementation may be enhanced also by automatic or manual clustering algorithms [CAN90, CROU89] which can be applied to nodes to generate default web link structures or node folders.

ASCII nodes can be made open and shareable by investigating algorithms that separate link anchors from node contents in a similar way to the method we suggested for screen nodes.

4. CONCLUSION

The re-usability concept is successfully applied in many areas of computer science (an example is the view concept in database systems) and is found to be very helpful in increasing productivity and reducing storage costs. In this paper we introduced a new model, the Node Re-usability model, for structured authoring and node sharing in hyperdocuments. The model consists of four layers: a Storage layer, a Link layer, and a high-level Presentation layer. The fundamental principle of the model is the complete and effective separation of link structures and node contents. The implementation details of a hypertext authoring system are demonstrated through a working prototype called HypAS which is based on the proposed model.

REFERENCES

[BUSH45] Bush, V. As we may think. *Atlantic Monthly.* 176, 1 (July 1945), 101-108.

[CAN90] Can, F., Ozkarahan, E. A. Concepts and effectiveness of the cover-coefficient-based clustering methodology for text databases. *ACM Transactions on Database Systems.* 15, 4 (Dec. 1990), 483-517.

[CONK89] Conklin, J., Begeman, M. gIBIS: A hypertext tool for team design deliberation. *ACM Transactions on Office Information Systems.* 6, 4 (Oct. 1988), 303-331.

[CROU89] Crouch, D. B., Crouch, C. J., Andreas, G. The use of cluster hierarchies in hypertext information retrieval. In *Hypertext '89 Proceeding.* (Nov. 1989), 225-237.

[ENGE68] Engelbart, D., English, W. A research center for augmenting human intellect. In *Proceedings of the AFIPS Fall Joint Computer Conference.* 33, (1968), 395-410.

[FRIS92] Frisse M., Cousins S. Models for hypertext. *Journal of the American Society for Information Science.* 43, 2 (Mar. 1992), 183-191.

[NELS67] Nelson, T. Getting it out of our system. In *Information Retrieval: A Critical Review.* G. Schechter, Ed. Thompaon Books, Washington, D.C., 1967.

[NIEL90] Nielson, J. *Hypertext and Hypermedia.* Academic Press, San Diego, CA, 1990.

[STOT89] Stotts, F., Furuta, R. Petri-net-based hypertext: Document structure with browsing semantics. *ACM Transactions on Information Systems.* 7,1 (Jan. 1989), 3-29.