Computer Science and Systems Analysis

Computer Science and Systems Analysis

Technical Reports

Miami University *Year* 1994

Structuring Hypertext Databases Using Hyper-Plate

Reginald Sequeira
Miami University, commons-admin@lib.muohio.edu

MIAMI UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & SYSTEMS ANALYSIS

TECHNICAL REPORT: MU-SEAS-CSA-1994-008

Structuring Hypertext Databases Using Hyper-Plate

Reginald Sequeira

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

Structuring Hypertext Databases Using Hyper-plate

by

Reginald Sequeira
Systems Analysis Department
Miami University
Oxford, Ohio 45056

Working Paper #94-008 Dec., 1994

Structuring Hypertext Databases Using Hyper-plate

Graduate Research

by

Reginald S. Sequeira

In partial fulfillment of
the requirements for the degree of
Master in Systems Analysis
Department of Systems Analysis

Miami University
Oxford, Ohio
December, 1994

Abstract

Hypertext is the non-sequential access of information. The hypertext structure is powerful enough to represent informal, semiformal, and formal data. The construction of a hypertext document is a complex and laborious task. In general, hypertext documents are constructed manually. The manual method of construction may produce unstructured documents. It has been shown that a poor document structure contributes to the lost in hyperspace syndrome. Hypertext Authoring System (HypAS) was developed at the Department of Systems Analysis, Miami University, Oxford, Ohio. In this study, Hyper-plate system was developed to structure documents created in HypAS. Hyper-plate system is used to create hyper-plates, i.e., templates for hypertext documents. A hyper-plate defines the attributes of a collection of nodes and the relationship among those nodes. The hyper-plate concept was conceived to address the issues of structuring and automating the construction of hypertext documents. The concepts of this study can be extended to any hypertext authoring system. The paper presents the design and implementation of Hyper-plate. The system components, data structures, operations and its integration with HypAS are described in detail.

Key words and phrases: hypermedia, hypertext, hypernodes, hyperlinks, HypAS, database, web, node reusability, hyper-engine, hyper-plate.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Hypertext is hailed as the future of information retrieval and dissemination. A hypertext document is simply a collection of linked nodes called *hypernodes*, containing any type of video or audio information [NIE90a]. The future hypertext systems will convey tactile and olfactory information to the user. In a hypertext document, relationships are defined among the nodes by linking them in various ways. This permits each user to customize a given hypertext document to suit his/her needs or preferences. The concept underlying the hypertext system, was described in 1945 by Vannevar Bush in his *memex*, a hypothetical hypertext engine [BUS45]. The term *hypertext* was coined by Ted Nelson in the late 1960s to describe Xanadu [NEL67], an electronic publishing system.

Current hypertext systems are essentially interactive systems, and the user retrieves related pieces of relevant information by following non-sequential paths called *hyperlinks* [NIE90a]. This method of retrieving related information by activating non-sequential links is termed as *browsing*, or *navigating* the hypertext document. The main interest in hypertext is centered on its usability as a vehicle for on-line interactive information retrieval and dissemination [CAN93a]. Recently there has been considerable interest in hypertext database systems.

CD-ROM is an attractive alternative to on-line telecommunication access for distributing very large commercial databases, and hypertext is the most appropriate data organization for effective CD-ROM based information retrieval [WES91]. Combining hypertext structure with database models facilitates the support of query language interface, views, and efficient storage

structures [HAR91]; while still retaining the hypertext's capability of fast access to relevant information. The major drawback of the hypertext approach is; as many as 56 percent of users lose orientation while browsing through a hypertext document [NIE90b]. This is termed as the *lost in hyperspace syndrome*. It has been shown that a poor hypertext document organization contributes to the lost in hyperspace syndrome [BER90]. The most preferred method to help the user traverse hyperspace without losing orientation is the provision of graphical browsers, backtracking facility, guided tour, and history mechanisms [NIE90a].

A graphical browser provides a visual representation of the hypertext document's graph structure [CON87], and facilitates the user in locating his current position in the hypertext network. Even though a browser is the most useful navigational aid, few systems provide this facility. The most widely used recovery approach after losing one's bearings in the hyperspace is the backtracking facility, or a jump to the beginning of the document (some systems call this the *home node*). Although, the lost in hyperspace syndrome is primarily due to the lack of a conceptual model during the hypertext creation process, the issue has not been properly addressed [NAN91].

There are two approaches to implementing a hypertext system. The stack (a Macintosh term) approach, which is used in HyperCard, stores the nodes and links in a single file. The second approach, the web approach, stores the nodes and links in separate files. Hypermedia, which was developed at Brown University [YAN88]; the World-Wide Web [VET94]; and HypAS, which was developed at Miami University [ABD93a]; are implementations of this approach.

In most current hypertext systems, an author creates each of the nodes, which contain information, and manually links them. This is a bottom up approach [ABD93b]. Evidently, a bottom up approach for hypertext document design or any information design should be avoided. It forces the designer to begin at the microscopic level first, and it is not flexible and results in wasted resources. What is required is a top down approach to hypertext document creation. This will allow one to look at the complex problem at a macroscopic level. It is flexible; since the design can be altered easily; and will be productive. An authoring tool such as an *auto template* or what we term as a *hyper-plate* addresses these issues.

We coined the word Hyper-plate, to designate the auto template system that we have developed to complement the HypAS hypertext system. There has been some interest in templates as an authoring tool. Catlin et.al.[CAT91] describe a Hypermedia Template tool developed at Brown University's Institute for Research in Information and Scholarship. In Hypermedia Templates, nodes and links are created when a Template is created. These nodes and links are stored in a folder, and a copy is made when the Template is instantiated.

We have a fundamentally different view of Templates, which we will call hyper-plates in the rest of this document. In our concept, a hyper-plate defines the attributes of a collection of entities (nodes) of a hypertext document and the relationship (links) between those entities [CAN93b]. A hyper-plate creates entities only when it is *instantiated.* We are primarily interested in hyper-plates as a tool to structure hypertext documents.

In section 2, we briefly discuss the fundamentals of hypertext. Section 3 gives a formal definition of hyper-plate and enumerates its advantages. We also demonstrate its usability to structure a hypertext database. In Section 4, we describe the Node Reusability model; and HypAS, a hypertext system based on that model. Section 5 is devoted to the description of Hyper-plate system, its components, interfaces, data structure, and operations. In section 6 we present conclusions and directions for future work.

2. FUNDAMENTALS OF HYPERTEXT

There are widely divergent views on the structuring of a hypertext document. At one extreme, Ted Nelson's "everything for any user" philosophy [NEL87] encourages liberal application of links by each and every hypertext user. The price is that the users are not guaranteed a clear structure; they are caught in a vicious circle in the hyperspace. At the other extreme is the rigid hierarchical structure proposed by Halasz [HAL87] which completely compromises the flexibility of the hypertext. We subscribe to a middle of the road philosophy, and therefore, conceived the hyper-plate concept to reflect this philosophy.

An author of a hypertext document should identify clear structural hierarchies prior to embarking on the creation of the document. Waiting until the document is complete is too late, since the author may not be able to see a clear structure. However, we permit the creation of more links between the nodes than those agreed on at the beginning. This will allow a small amount of flexibility to the author to make the document more interesting, and to take care of unforeseen situations.

2.1 Document Structures and Components

The structure of a hypertext document is a directed graph or a di-graph. In a circular hypertext, a user will reach the initial node after following every node in the di-graph's path as in Figure 1(a). There is no order for reading; it does not matter from which node the reading starts. In a fully connected hypertext, represented by Figure 1(b), every node is connected to every other node [BOT91]. The readers have no clue as to which article should be read next. A tree structure as in Figure 2, has well-defined hierarchies. The nodes and links of a hypertext structure have *types*.

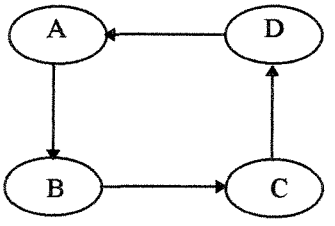

Figure 1(a)
Circular hypertext structure.

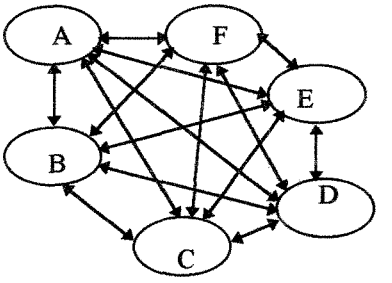

Figure 1(b)
Fully connected hypertext structure.

2.1.1 Node Types

The nodes in a hypertext document have the information content. Usually they have textual information. They may also have animation or audio information. They are of the following types.

Root Node

The fundamental property of a root node is that it has to reach every node in the hypertext document. Another property of the root node is that the distance from it to every other node should be small [BOT91]. It is usually the point of entry into the document. Sometimes it is

also referred to as the *home-node*. Usually a mechanism is provided to return to the root node if the user loses orientation. In Figure 2 University is the root node.

Index nodes

Index nodes are nodes that can be used as guide to many other nodes. Therefore, they will have several outgoing links. For example, in a hypertext document on American history, a node containing an article on Founding Fathers would point to all the nodes that have information on Founding Fathers.

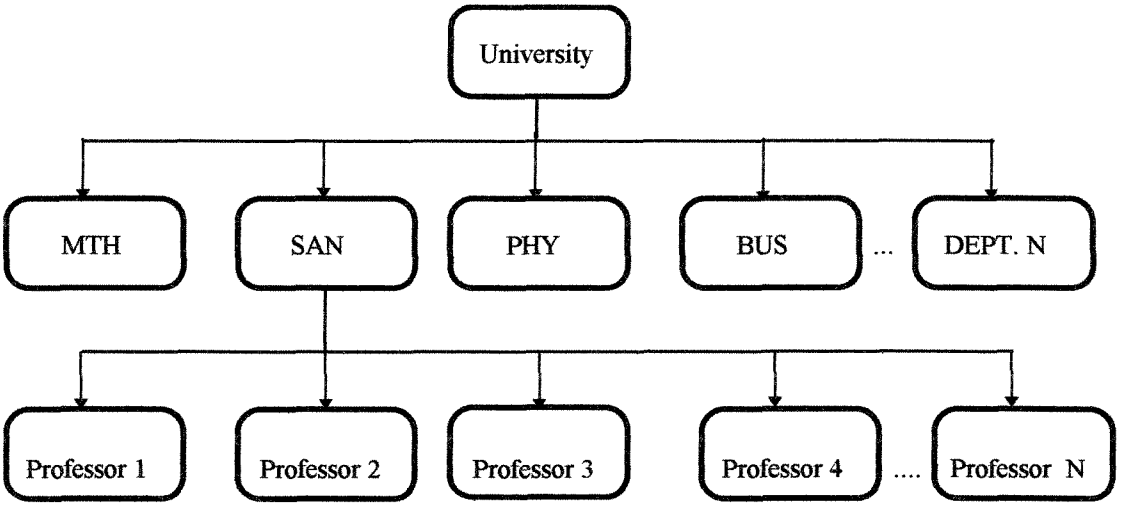

Figure 2. An example for regular and repeating hierarchical hypertext structure.

Reference Nodes

Reference nodes are pointed by several nodes and will have several incoming links. All the Founding Fathers will have a reference to the Declaration of Independence. Therefore, a node containing the article on Declaration of Independence would be a reference node.

Open nodes

They are also called unconnected nodes. They do not have any links pointing to them, and they cannot be accessed from the root node by traversing along any path. A hypertext document with such nodes will not provide complete information. There will be discontinuity in the flow of information.

2.1.2 Link Types

A link connects two nodes in a hypertext structure, hence it defines the relationship between those two nodes. A link originates from an anchor (source) node and terminates in a target node. A button is a block of text or icon in the anchor node. A button activates the link connecting that node to the target node. Links have types.

Uni-directional Hypertext Links

These links permit the user only to jump to a target node from an anchor node. They allow the author to impose a clear hierarchy on the document. They also permit the document to be well structured so that there is a natural flow to the content.

Bi-directional Links

These links permit the user to access the target node and anchor node by traversing along the same path. They are required; since, an anchor node and a target node may have content that references each other.

Dangling links

These are links that have an anchor node but do not have a target node. They are the result of deleting a node without taking proper care to delete all the links which point to that node.

They have a deleterious effect on the hypertext structure and contribute to the lost in hyperspace syndrome.

2.2 Hypertext Document Creation Process

The construction of a hypertext document is a laborious and time consuming process. Here, we describe the process of constructing a hypertext document on HypAS. It is similar to the process of constructing a hypertext document on any other system using the web concept. The author of the document creates individual nodes one at a time using the hypertext system editor and stores them on the disk as files. When a sufficient collection of nodes is built up, one node at a time is loaded into the system and links are forged between the different nodes. Then more nodes are created and added to the document. Apart from the technical aspects, the hypertext document creator should have the knowledge of information design principles, such as guidelines for formatting documents according to a specific style [CAT91]. The creation of a well-designed document requires the expertise of people in two different disciplines: the information designers and the content experts. Should the information designer build the whole system, the result will be an uninteresting and uninformative system at best, or inaccurate information at worst. If the content expert tries his hand at building nodes and links, the resulting system will be a maze of nodes and links with no discernible structure, several dangling links, and open nodes. For example, to construct a hypertext document of marine life, a marine biologist will be the content expert and an information designer will have the knowledge to present this information in a most informative and effective way.

3. HYPER-PLATE: A SOFTWARE TOOL FOR HYPERTEXT SYSTEMS

Hypertext system's flexibility makes them suitable for wide ranging applications: computer, business, intellectual, educational, entertainment, and leisure [NIE90a]. However, the widespread acceptability of hypertext systems (or any computer application) is contingent on the ease with which they can be used. The difficulty of creating a hypertext document should be addressed more aggressively. Currently, a certain level of expertise is required to create a small hypertext document, and the creation of a large hypertext document is indeed a complex and laborious task. During the creation of a hypertext document having as few as five nodes *the author* can be lost in the hyperspace.

3.1 Definition of hyper-plate

A hypertext template or a *hyper-plate,* T, defines a set of nodes Nt and a set of relationships Rt between the nodes. The set of relationships is represented by the set of links Lt. The hyper-plate can be represented by the pair $T = \langle Nt, Lt \rangle$. A single hyper-plate is instantiated several times such that:

$$\textit{for } Ti \textit{ and } Tj,\ Ni \cap Nj = \varnothing \textit{ and } Li \cap Lj = \varnothing \textit{ for all } i, j \in I\ (i \neq j),$$

where I is the index set for all instances.

A web W is a pair $W = \langle N, L \rangle$

Where N = is the set of nodes in the web and,

L = is the set of links in the web.

When we use a hyper-plate T, to create a web W,

N is a family of node set $\bigcup i \in I\, Ni = N$ and

L is a family of link set $\bigcup i \in I\, Li = L$.

3.2 Hypertext Documents from hyper-plate

In a Hyper-plate environment, an information design expert constructs the *Master hyper-plate* and stores it. A user or the content expert *instantiates* the master hyper-plate. The master hyper-plate is not destroyed; it is saved for future instantiations. When a hyper-plate is instantiated, a unique set of nodes and a unique set of links are generated. All new instances inherit the properties; such as color, configuration, formatting information, contents, and linking information from the master hyper-plate. Figure 3 is a master hyper-plate for the entity professor. After instantiation, the user fills the new instance with appropriate information. Several instances can be generated using one master hyper-plate. The system can store several hyper-plates designed for different applications. We can also have several hyper-plates for general applications such as; instructional, demonstration, and presentation purposes. The user can view each of these hyper-plates and select the best one that suits his/her needs.

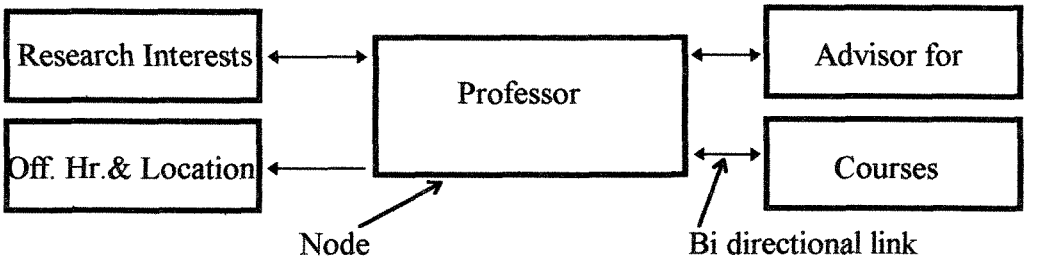

Figure 3. An example hyper-plate.

3.3 Advantages of hyper-plates for Hypertext Document Creation

A hyper-plate facility can alleviate many of the problems associated with the creation of a hypertext document. It is a software tool designed to increase productivity, decrease the waste of resources, and will have the following benefits:

- It makes the use of hypertext systems attractive to novice users;
- It enables the creation of well-structured hypertext documents by inexperienced users. The author does not have to know to create links and nodes;
- It completely automates the creation of hypertext documents;
- It facilitates the application of information design principles in the construction of hypertext documents;
- It presents a clear picture of the hypertext structure, that is, the interconnection among the nodes to the user;
- It eliminates the presence of dangling links and unconnected nodes in the document;
- It imposes a pre-determined structure on the document; thus it eliminates the possibility of creating a document with undesirable structure.

3.4 Do We Need hyper-plates?

The natural question that comes to mind is why should we use a hyper-plate to create the set of nodes and links in a repeating hierarchy. Why not create a first set of nodes and links, link them into a web, and then create multiple copies of that set of nodes and links? Let us consider the repeating hierarchy of Figure 2. Prof. A has created a set of nodes and linked them into the department web. Prof. B, makes a copy of those nodes, deletes all the information content, and fills the nodes with new content. His intention is to create a web as in figure 4. However, when Prof. B tries to access the information which he thinks is his, he will be still accessing the information belonging to Prof. A. The problem here is that, even though, Prof. B made a new set of nodes, the linking information in the web he copied still links the

node Prof. B to the set of nodes {course A, office A, student A, AI}. The actual result is illustrated in Figure 5.

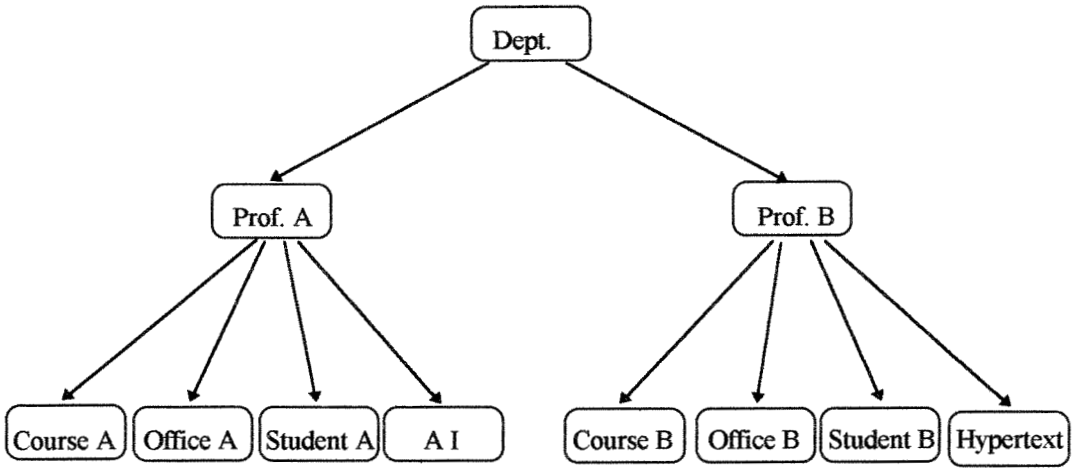

Figure 4. Prof. B intends to create a hypertext document by copying Prof. A's nodes.

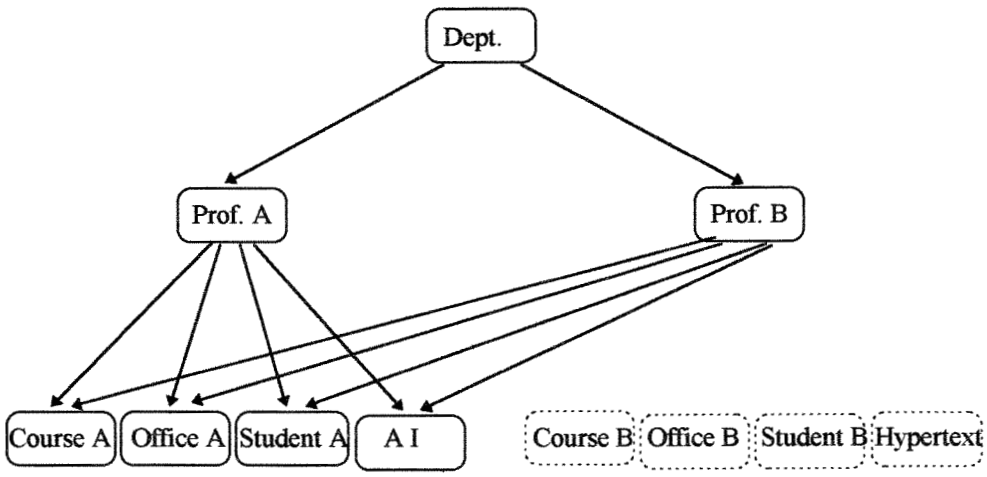

Figure 5. The result is not what Prof. B had intended. He accesses Prof. A's nodes instead of his.

3.5 Hypertext Databases

Hypertext systems are most suitable to represent non-linear information, very much like the human mind. The basic hypertext structure itself is powerful enough to represent structures

ranging from formal (Database tables), semi-formal (semantic networks) to informal (unstructured data). However, since, hypertext structure is primarily intended for representing informal structures, developing formal structures is a difficult task [JOR89]. For example, the development of an employee database involves repetitive creation of entities, and links between the entities (e.g., employee and department). Realizing such a database in hypertext will require repetitive creation and organization of nodes and links.

3.5.1 Structured Hypertext Databases

From the database theory point of view, a hypertext web can be considered as a *view* in the hypertext model [UTT89]. However, the manual construction of a structured hypertext database is expensive because of the high cost, or cognitive overhead [CON89]. It also does not assure consistency and coherency in such a database. If directly linked nodes contain related information then they are coherent. We propose that a hyper-plate is the *schema* for the web, i.e., from the hypertext point of view it is equivalent to the schema of the relational database. Just like a schema, a hyper-plate does not create entities by itself; it merely defines the attributes and relationships of the entities that will be created at the time of instantiation. The state of the department web after three instantiations of the Prof. hyper-plate is shown in Figure 6.

3.5.2 Unstructured Hypertext Databases

It may appear that hyper-plates are inapplicable while creating a hypertext version of the encyclopedia, which is unstructured data from the database perspective. However, the

information design principles require that the structure of the document be represented on paper first, perhaps using drawing software package. It is also desirable to breakdown the full information into small blocks of data, and organize them into a small number of nodes that can be easily managed [ABD93a]. These small blocks of nodes can be directly represented on the Hyper-plate's editor instead of using drawing software package. This would not only automate the creation of the document itself, it would also eliminate one step in the creation of the document. One hyper-plate for an individual block of information can be created, and finally the document can be assembled by instantiating the collection of hyper-plates belonging to this document.

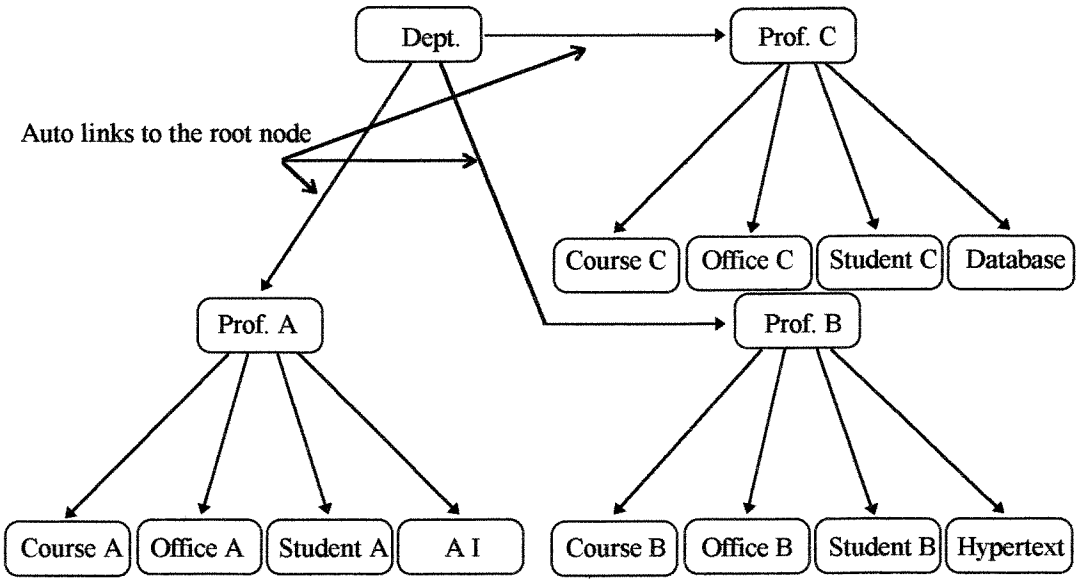

Figure 6. A department hypertext document created using hyper-plate.

4. THE NODE REUSABILITY MODEL AND HypAS

We elaborate on the two approaches currently used for the implementation of hypertext systems. In the first approach(used in HyperCard), the nodes and links that constitute a hypertext document are stored in a single file. If some node of this document contains

information that may be of interest to another hypertext document, then a copy of that node is made, and that copy linked appropriately in the new document. This is a duplication of information and waste of storage space. We point out that to avoid duplication just creating links from the second document to the first will be inappropriate. The links intended for the first document will be accessible from the second document also. In this approach we cannot present different views of the same information effectively. Different views of the same information are required to present the same information to different users of the document.

The second approach stores the nodes and linking information separately. This approach is used in the Node Reusability Model (NRM), which is employed in the implementation of HypAS. A similar approach is also used in Intermedia [YAN88]. In NRM each node is a separate entity. To construct a document, the associated nodes are linked to form a network of nodes called the *web* and the web is stored in a separate file. A hypertext document can be a single web, or a superweb: a set of webs. Since the linking information is not part of the nodes, it is possible to include a given node into different webs without being able to access one web from another. The advantages of the node reusability model are:

- A single node can be utilized in different hypertext documents without duplicating the node;
- Nodes can be authored independently;
- New nodes can be added to the collection of nodes, without incorporating them in any document;

- Update anomalies are eliminated, since information is updated in only one node and this propagates to all webs that access this node;
- The same information can be presented from different perspectives by developing more than one web.

4.1 HypAS

HypAS is based on the Node Reusability Model. 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 [ABD93b]: a user interface subsystem, a nodes subsystem, a links subsystem, and files and storage subsystem corresponding to the layers of NRM (see Figure 7). HypAS was developed in the Department of Systems Analysis, Miami University, Oxford, Ohio. It is an easy to use system and it is intended to be used for research and instructional purposes [ABD93a].

Two distinct user interfaces are available in HypAS. One is for the hypertext reader, and the other for the hypertext author. The authoring interface is the HypAS editor, which is used for creating and editing nodes, links, and webs. In HypAS nodes can be of four types: screen, index, command, and script. Screen nodes are text nodes that can be created using the HypAS editor. In HypAS, any MS-DOS program (.EXE or .COM) or batch file (.BAT) is qualified to be a command node. This allows the user to link nodes that may contain graphics, animation, digitized sounds, etc. HypAS supports a superset of Basic language for the creation of script nodes. In HypAS, each web may contain a maximum of fifty nodes. Webs are inter-linked to

form a superweb, and up to ten webs may be contained in a superweb. The system can support a total of five hundred superwebs. Very large hypertext document can be created [ABD93a], since the system can contain 250,000 nodes. Each web is stored under a unique name. For the reader, HypAS provides a list of superweb files to select from. The reader can select one superweb for loading into the memory. When a superweb is selected, the network of link structures for all its web members are loaded into the main memory. HypAS supports backtrack facility as a recovery mechanism.

5. AN IMPLEMENTATION OF HYPER-PLATE FOR HypAS

The Hyper-plate system for HypAS, is implemented in the C++ language to ensure total compatibility with the existing system. It runs on any PC using the MS-DOS 3.0 or later versions. It has three layers: the top most layer is the user interface system, the bottom layer is the file storage system, and between the two is the Hyper-plate engine or the *hyper-engine*. In this section we describe each layer in detail.

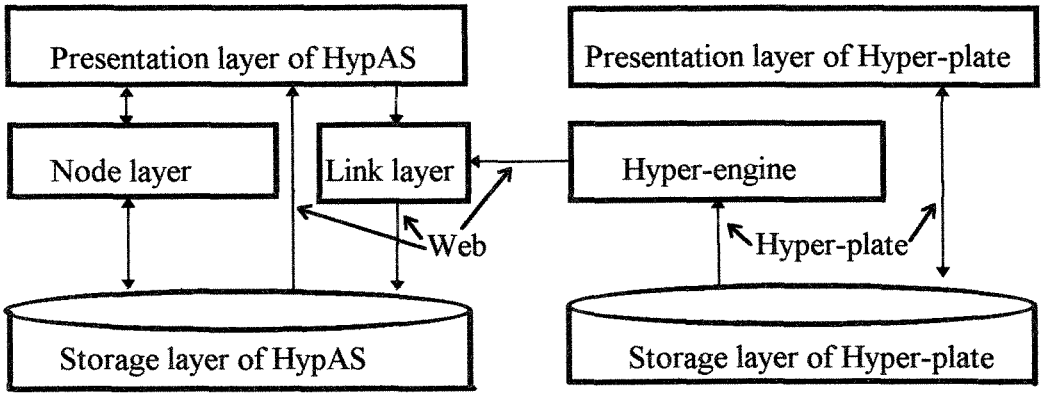

Figure 7. Interconnection and data flow between the layers of HypAS and Hyper-plate.

5.1 The User Interface System

Four distinct user interfaces are provided by the Hyper-plate system. First, the Hyper-plate editor is for the designer of the hyper-plates. There are three interfaces for the user or the author of the hypertext documents: the Hyper-plate viewer, the node editor, and the screen editor. The interfaces are designed with a view to automate most of the tasks. The designer or user can accomplish the job with minimal technical knowledge. We predict a very short learning period for the user of the Hyper-plate system.

5.1.1 The Hyper-plate Editor

It is a menu driven user friendly graphical editor. The user can depict four types of objects on the desktop: root node, unique nodes, uni-directional links, and bi-directional links. The root node is represented by double bordered rectangle, unique nodes by single bordered rectangles, uni-directional links by unidirectional arrows, and bi-directional links by bi-directional arrows. The editor recognizes these entities as node objects and link objects. Root node is created only once, during the first instantiation of the master hyper-plate.

Unique nodes are created during every instantiation of the hyper-plate. They are called *unique* because only one instance of the hyper-plate can access them. Root nodes and imported nodes can be accessed by every instance of the hyper-plate. Obviously, every node is unique from the web point of view. A node object can be drawn by specifying the location of the object on the screen and the type of object; that is, whether unique or root node. Command and script nodes are also depicted with double bordered rectangles. A double bordered rectangle merely

specifies that a node should be created only once, during the first instantiation. The arrows representing the links are drawn automatically once the user specifies the nodes to be linked. Deletion of nodes is accomplished by simply placing the cursor inside the rectangle representing the node and selecting delete. When a node is deleted, all links associated with that node are also deleted. This prevents the presence of any dangling links in the document. There is provision for the manual creation and deletion of links, as well as for the renaming of nodes.

A Hyper-plate editor can be used to create a new hyper-plate or edit an existing one. A hyper-plate can be renamed. However, renaming a hyper-plate will save it as another hyper-plate. For example, if we rename a Prof. hyper-plate to Employee, we will have two hyper-plates; one for Prof., the other for Employee. After creation, hyper-plates are stored in the system.

5.1.2 The Hyper-plate Viewer

We mentioned that a system can have several hyper-plates. A hyper-plate viewer is a facility that lets the user get a list of hyper-plates in the system and view them one by one. A hyper-plate is not instantiated while it is viewed, neither can it be edited nor modified.

5.1.3 The Node Editor

This interface instantiates a hyper-plate and opens the new instance to the user. The user can open each node by placing the cursor on the node and selecting open. The node editor then searches the file storage system for that node. If a node does not exist, it creates a new one,

and loads the node into the screen editor. An opened node is marked by a different color to indicate that it has already been edited.

5.1.4 The Screen Editor

This interface allows the user to edit the nodes accessed from the node editor. When a node is opened, the system automatically starts the screen editor and displays the node in it. The node title is displayed at the top of the screen, and the links at the bottom. The author can then fill the node with content and create extra links to other existing nodes. This feature allows the creation of links that could not have been foreseen. In addition, the author can draw objects on the screen, move objects, and change the color of the text. The information is saved automatically when the user exits the screen editor.

5.2 *Hyper-Engine*: The Hyper-plate Engine

Hyper-engine is the heart of the system. The input to the hyper-engine is the output from the Hyper-plate editor. When a hyper-plate is instantiated, the user is given the option of either creating a new web, or to incorporate the new instance of the hyper-plate into an exiting web. If the user requests the creation of a new web, the hyper-engine converts the information contained in the hyper-plate data structure into a web. If the user requests the hyper-plate to be incorporated into an existing web, a list of webs is displayed first. After a web is selected a list of hyper-plates is presented. The hyper-engine then loads the web first into the system; converts input from the hyper-plate into hypertext nodes and links, and updates the web. The hyper-engine keeps a count of all the copies it has instantiated for each hyper-plate. When a

hyper-plate is instantiated, it generates a unique file name for each node, and links it with the correct set of links.

5.3 The Files and Storage System

The files and storage system contains the routines for creating, storing, and accessing the files created by the hyper-engine and the user interfaces. The files created are hyper-plate files, screen nodes, and web files.

5.4 Integration of Hyper-plate into HypAS

The Hyper-plate system is integrated into the existing HypAS system at two layers: the presentation layer and the link layer. The access to Hyper-plate is through the Main menu of HypAS, a presentation layer. Also, during the editing and linking of the node object, it makes use of some of the existing HypAS presentation layer functionality. At the link layer, it takes control of the HypAS web when Hyper-plate is being used. A web created by using Hyper-plate can be loaded, edited, and run from HypAS. We can create more links from HypAS if desired. HypAS can also be run without the Hyper-plate software.

5.5 Hyper-plate Data Structures

Hyper-plate has three important data structures of its own. It also utilizes the existing link, node, and web data structures of HypAS during the instantiation time. In this section we describe the Hyper-plate data structures. For a detailed description of HypAS data structures the reader is referred to the working paper of Abdalla [ABD93b].

5.5.1 Link Object Structure

Hyper-plate recognizes links and nodes that are represented on the Hyper-plate editor. The link objects contain the linking information that is later used to create hyper-links at the instantiation time. Their structure is as in Figure 8.

```
typedef struct {
        boolean flag;
        int anchor [2];
 /* x, y positions of anchor point on the desktop */
        int target [2];
 /* x, y positions of target point on the desktop */
        char nodes [2] [NODE_NAME_SIZE];
 /* anchor and target nodes */
   } link_obj_type;
```

Figure 8. Record structure of the link object.

5.5.2 Node Object Structure

The node objects contain their position on the editor desktop and an object name. If an object of that name does not exist in the file storage system, an object with a unique file name is generated during instantiation. Node object structure is as in Figure 9.

```
typedef struct {
        boolean flag;
        char obj_name [NODE_NAME_SIZE];
        char obj_type;
 /* unique or non- unique type of node object*/
        int top [2];
        int bottom [2];
 /* x, y positions on the desktop */
   } node_obj_type;
```

Figure 9. Record structure of the node object.

5.5.3 Hyper-plate Record Structure

The hyper-plate record contains information about all the node objects and link objects that are on the desktop. The desktop is saved as a screen file. The hyper-plate record also stores the count of instances it has generated. The record structure of hyper-plate is in Figure 10.

```
typedef struct {
        char temp_name [FILE_NAME_SIZE];
        char instance [INS_SIZE];
        int web_ins_point;
  /* insertion point in the web */
        obj-type node [MAX_NODES];
        link_type link [MAX_LINKS];
  } template_type;
```

Figure 10. Record structure of the hyper-plate.

5.6 Hyper-plate Operations

The presentation layer and hyper-engine have their own set of operations. The layers and their operations are modularized. The functions are cohesive and operate strictly by parameter passing. Following are the lists of important operations and a brief description of each.

5.6.1 Presentation Layer Operations

Create_temp (Template_type)

Creates a hyper-plate with no objects in it. Prompts the user for a name for the hyper-plate, if there is no user input, assigns a default name for the hyper-plate.

Open_temp(Template_type)

Gives a list of currently available hyper-plates to the user. Loads an existing hyper-plate from storage, and opens it for editing.

Save_temp(Template_type)

Saves a hyper-plate to the disk as a data file. The user can change a hyper-plate file name during this operation.

View_temp(Template_type)

Gives a list of available hyper-plates and accesses the specified hyper-plate from storage and displays it.

Create_node(Template_type, Node_name, Node_type)

Creates a representation of the node (rectangle) on the editor screen, and stores the object information in the hyper-plate record.

Rename_node(Template_type, Node_name, Node_type)

Changes the name of a node and modifies the information in the hyper-plate record.

Access_node(Node_name)

Access the node file from the disk if it already exists, otherwise creates a new node. Loads the screen editor and the node into the screen editor.

Save_node(Node_name)

Saves a screen node to the disk.

Delete_node(Template_type, Node_name, Node_type)

Deletes the node representation from the desktop, deletes all the links associated with the node, and erases the node and link information from the hyper-plate record.

Create_link(Template_type, link_type)

Automatically displays an arrow representing a link between two nodes, and stores the information in the hyper-plate record.

Delete_link(Template_type, link_type)

Deletes a link and deletes this link's information from the hyper-plate structure. Also, erases the arrow representing this link from the desktop.

5.6.2 Hyper-engine Operations

Make_web(Template_type, Web_type)

Creates a web from the hyper-plate record , and loads it into the system.

Load_web(Template_type, Web_type)

Loads an existing web from the disk storage, updates it with new node and link information generated from the hyper-plate record.

Save_web(Web_type)

Saves a web after the new instance of the hyper-plate has been edited.

Update_Temp(Template_type)

After every hyper-plate is instantiated, the information that a new instance has been created is stored in the hyper-plate's record.

6. CONCLUSIONS

Hypertext is the non sequential retrieval of information. However, this non sequentiality can lead to confusion if used indiscriminately during the authoring phase. A basic hierarchical hypertext structure, with some cross referential links is most likely to lessen user

disorientation. Constructing a hypertext document is a laborious and repetitive task especially when regular hierarchies are involved. A hyper-plate, for hypertext database, is equivalent to a schema of the relational database. In addition, it has the advantage of automating the construction of hypertext documents. The Hyper-plate system implemented for HypAS has three layers: a storage layer, a presentation layer, and the hyper-engine.

Hypertext is suited to on-line information. There are still issues to be addressed in hypertext databases. The issues of insertion, deletion, and update anomalies should be addressed. In hypertext, these anomalies are caused by missing or dangling links. Like the schema of the relational database, hyper-plate cannot address these issues. We suggest the development of a *smart-web* to overcome the current drawbacks. Such a web would automatically detect dangling links.

In our investigation for this paper, we discovered that the research in hypertext is concentrated on the analysis of the hypertext structure itself. We have not found any attempt to evaluate the hypertext systems and quantify the results. The current method of evaluation is just user feedback which is subjective. Another area to explore could be the evaluation of hyper-plate vis-à-vis the manual method of constructing a hypertext document. However, sizable user participation will be required to carry out these evaluations.

REFERENCES

[ABD93a] Abdalla, O.,Can, F. "Node Re-Usability in Structured Hypertext Systems." *In Proceedings of the ACM 1993 Computer Science Conference*, 1993,ACM, New York, 440-445.

[ABD93b] Abdalla, O. "Design and Inplementation of a Hypertext Authoring System (HypAS)." Working paper #93, Miami University, Oxford, Ohio, 1993.

[BER90] Bernstein, M. "Hypertext and Technical Writing." *In Proceedings ECHT '90 Course*, Versailles (France), 1990.

[BOT91] Botofogo, R. A., Shneiderman, B. "Identifying Aggregates in Hypertext Structures." *In Hypertext '91 Proceedings*, 1991, 63-74.

[BUS45] Bush, V. "As we may think." *Atlantic Monthly*. 176 (1), July 1945, 101-108.

[CAN93a] Can, F., Lee, Y-M. "HypIR: A Hypertext-Based Approach to Information Retrieval." *In Proceedings of ACM 1993 Symposium on Applied Computing*, 1993, New York, 729-736.

[CAN93b] Can, F., Sequeira, R.S. "Structured hypertext creation using templates: Auto template tool for HypAS." *Proceedings of the Eighth Int. Symp. on Computer and Information Sciences*, Istanbul, Nov. 1993, 549-556.

[CAT91] Catlin, K. S., Garret, L. N. "Hypermedia Templates: An Authors Tool." *In Hypertext '91 Proceedings*, 1991, 147-160.

[CON87] Conklin, J. "Hypertext: An Introduction and Survey." *Computer*, 20(9), 1987, 17-41.

[CON89] Conklin, J., Begeman, M. L. "gIBIS: A Tool for All Reasons." *Journal of the American Society for Information Science*, 40(3), pp 200-213, May 1989.

[HAL87] Halasz, F.G., Moran, T.M., Trigg, R.H., "NoteCards in a Nutshell." *In Proceedings of the ACM CHI+GI Conference*, Toronto, April 1987, 45-52

[HAR91] Hara, Y.,Keller, A.M., Wiederhold, G. "Implementing Hypertext Database Relationships through Aggregation and Exceptions." *In Hypertext '91 Proceedings, 1991*, 75- 90.

[JOR89] Jordan, D. S., Russell, D. M., Jensen, A. S., Rogers, R. A. "Facilitating the Development of Representations in Hypertext with IDE." *In Hypertext '89 Proceedings*, 1989, 93-104.

[NAN91] Nanard , J., Namard, M. "Using Structured Types to Incorporate Knowledge in Hypertext." *In Hypertext '91 Proceedings*, December 1991, 329-343.

[NEL67] Nelson, T. H. "Getting it out of our system." *In information Retrieval: A Critical Review*. G. Schechter, Ed. Thompson Books, Washington, D.C., 1967.

[NEL87] Nelson, T.H. "All for one and one for all (invited to talk)." In F. Halasz (Ed.), *In Proceedings of Hypertext '87*, 5-7, Chapel Hill, NC: ACM Press.

[NIE90a] Nielsen, J. *Hypertext & Hypermedia*. Academic Press, Inc., 1990.

[NIE90b] Nielsen, J. "The Art of Navigating Through Hypertext." *Communications of the ACM*, 33 (3), March 1990, 296-310.

[UTT89] Utting, K., Yankelovich, N. "Context and Orientation in Hypermedia Networks." *ACM Transaction on Office Information Systems*, 7(1)1989, 58-84.

[VET94] Vetter, R.J., Spell, C., Ward, C., "Mosaic and the World-Wide Web." *Computer*, Oct.'94, 49-57.

[WES91] West, J. C., "Economic Constraints in Hypertext." *Journal of the American Society for Information Science*, 42(3), 1991, 178-184,

[YAN88] Yankelovich, N., Haan, B.,Meyrowitz, N. K., Drucker, S. M. "Intermedia: The Concept and Construction of a Seamless Information Environment." *Computer*, 21(1), 1988, 81-96.

APPENDICES

Appendix A

Hyper-plate User's Guide

TABLE OF CONTENTS

1. About Hyper-plate

Hyper-plate is a system designed to automate the creation of hypertext documents in the HypAS environment. It is designed to impose clear structure, and solve many of the problems inherent to the creation of hypertext documents. Note that the output of the system "Hyper-plate" is also called hyper-plate. The context in which the term is used should clarify which one is referred. Also, the system is spelt with a capital 'H' and the system output with 'h.'

Hyper-plate was developed at the Department of Systems Analysis, Miami University, Oxford, Ohio. It is integrated into Hypertext Authoring System (HypAS) which was also developed at the Department of Systems Analysis, Miami University, Oxford, Ohio. For further information on HypAS and the user's guide on HypAS, please refer to the working paper of Abdalla [ABD93a]. A copy of this paper may be obtained from the Department of Systems Analysis, Miami university, Oxford, Ohio.

2. Installing Hyper-plate

Put the installation disk in drive A: (or B:), type **a: install** (or b: install) and press the <ENTER> key. The installation program creates a directory, C:\HYPAS, and installs the program in that directory. Since Hyper-plate is a part of HypAS, you will get an updated version of HypAS. If you have an older version of HypAS and want to save it, you should copy it to some other directory.

3. Starting HypAS and Hyper-plate

If you want to start HypAS always from the C:\ prompt, you should modify your AUTOEXEC.BAT file. Include C:\HYPAS in the "path" command of your AUTOEXEC.BAT file. If you do not want to modify your AUTOEXEC.BAT, change the directory to C:\HYPAS and type "hypas" every time you want to execute the program. You will be greeted with the HypAS greeting screen. The greeting screen will be displayed for five seconds; after that, the Main menu and the main screen of the HypAS will be displayed. The main screen consists of 23 lines of workspace area called the desktop. The bottom two lines of the screen are the Main menu and message area.

4 Using the Menus

HypAS and Hyper-plate are menu driven user friendly systems. The Main menu is always displayed first (see Figure A.1). The current selection is highlighted with a menu bar. You can change menu selection highlighting by using the <LEFT> or <RIGHT> cursor keys. Also, the spacebar moves the highlighting to the right. When a menu selection is highlighted, the next options available under that selection are displayed in the message area. Menu selection is executed by highlighting a selection and pressing the <ENTER> key. In some menus, selection can be also made by entering the first letter of the selection. If a menu offers that feature, then the first letter of the selection is displayed with a different color.

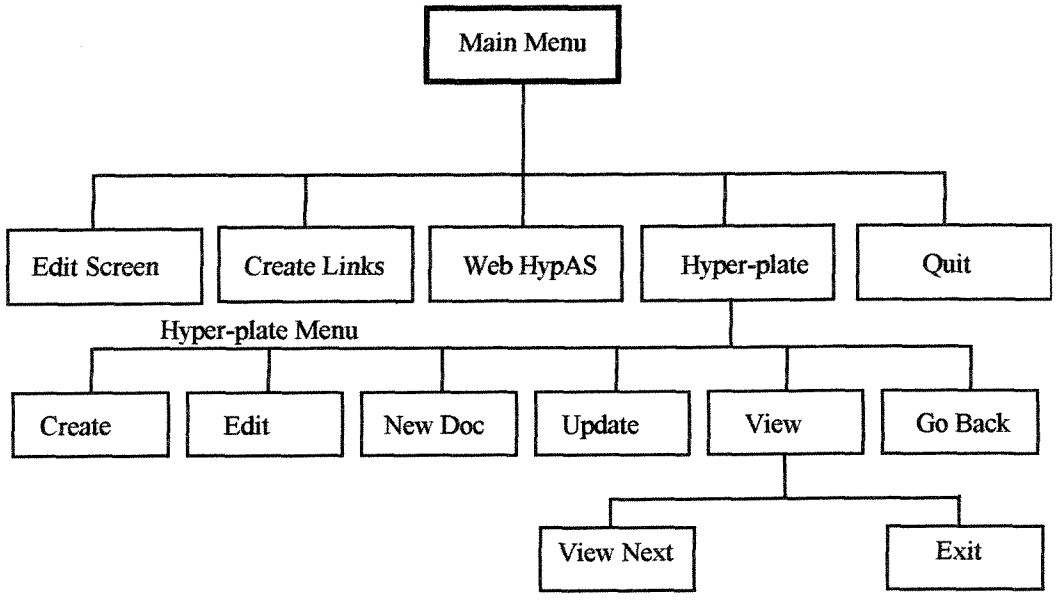

Figure A.1. Accessing the Hyper-plate Menu and View Menu from the Main Menu.

5. Hyper-plate Menu

From the Main menu of HypAS select "Hyper-plate" and press the <ENTER> key. Hyper-plate menu provides access to the Hyper-plate editor, Node editor, and the View menu. From the Hyper-plate menu you can go back to the Main menu by selecting "Go Back" and pressing the <ENTER> key.

5.1 View Menu

When "view" is selected from the Hyper-plate menu, a list of hyper-plate files that are in the system is displayed. Hyper-plate files can be then viewed one by one in the viewing screen by selecting "View Next". Selecting "Exit" while in the viewing screen, will bring you back to the Hyper-plate menu.

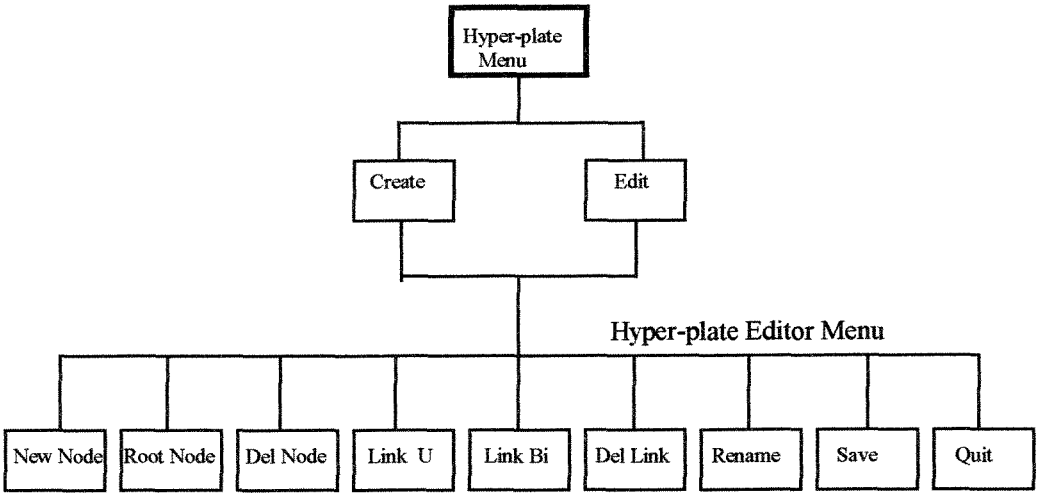

Figure A.2. Accessing the Hyper-plate Editor and the Hyper-plate Editor Menu from the Hyper-plate Menu.

6. Hyper-plate Editor

Hyper-plate editor can be accessed from the Hyper-plate menu by selecting either "Create" or "Edit" and pressing the <ENTER> key (Figure A.2). Hyper-plate editor is character based. The first 23 lines constitute the desktop. The bottom 2 lines show the menu selection. When "Create" is selected, you are prompted to enter a name for the new hyper-plate. The name can be up to eight characters long. All hyper-plates will have the extension ".TEM" appended to their file name. If no name is entered a default name (NONAME.TEM) will be assigned. You can cancel the selection by entering the <ESC> key. When "Edit" is selected, you are prompted to type a hyper-plate file name. You can also get a list of hyper-plate files that are in the system by simply pressing the <ENTER> key when prompted to enter a hyper-plate file name. A file can be then accessed by highlighting it and pressing the <ENTER> key again.

6.1 Hyper-plate Editor Menu

When you first access the Hyper-plate menu, the first selection on the menu will be highlighted. You can change the menu selection highlighting by using the <LEFT> cursor key, <RIGHT> cursor key, or the space bar. Any executed selection can be canceled by using the <ESC> key. The cursor will appear in the desktop area after a selection has been executed, and none of the menu selection will be highlighted. To make another menu selection press the <ESC> key, and once again the first menu selection will be highlighted. You can change the highlighting again as explained earlier.

6.2 Creating a node on the Hyper-plate Editor

Select the option "New node" or "Root Node" and press the <ENTER> key. You will be prompted to move the cursor to a position where you want to place the node and press the <ENTER> key again. You will have to then move the cursor by using the cursor keys to select the area of the rectangle that represents this node. After you have selected the area, press the <ENTER> key once more. The editor draws a rectangle to represent the node. The rectangle will be single bordered or double bordered depending on the type of node you have selected. The system then prompts you for the name of the node. If you do not enter any name, the system gives an error message and the rectangle is erased from the desktop. If you type a name, this will be displayed in the center of the node. The operation can be canceled at any stage by pressing the <ESC> key. Be sure to draw a rectangle large enough to enclose the name of the node. Selecting a smaller area will not affect the information stored in the hyper-plate, but it will not look legible on the desktop.

6.3 Deleting a node from the Hyper-plate Editor

To delete a node, select the option "Del Node" from the menu. Place the cursor inside the rectangle representing the node and press the <ENTER> key. If you place the cursor where no rectangle exists, the system gives a warning. The operation can be canceled by pressing the <ESC> key.

6.4 Creating links on the Hyper-plate Editor

Select either "Link U" or "Link Bi" and press the <ENTER> key. The system then prompts you for the names of the anchor and target nodes. If you enter an invalid name, the system gives an error message and does not create a link. If you enter valid names, the editor asks whether you want the system to create the link for you. If you type in "Y," which is "yes," the system creates an arrow to represent the link and asks whether this link is "O.K." If you enter "N," which is "no," the system erases the arrow and prompts you to create the link (arrow) manually. You can cancel the operation at any stage by pressing the <ESC> key. Note that the system can only create either horizontal or vertical arrows automatically. Also, there should be no objects between the anchor and target nodes. If there are, they it will be overwritten. However, it will not affect the information stored in the hyper-plate record. Uni-directional link will be represented by an arrow, with arrow head towards the target node. Bi-directional link will be represented by an arrow, with an arrow head on both ends.

If you select "N" when the editor prompts whether you want automatic link creation, the system then prompts you to create the link manually. To create a link manually, simply place

the cursor outside the rectangle representing the anchor node and press the <ENTER> key. Then move the cursor to the target node and press the <ENTER> key again. The operation can be canceled by pressing the <ENTER> key. Since the HypAS and Hyper-plate use character based editors, only horizontal or vertical arrows can be drawn. You cannot draw arrows at an angle to the vertical or horizontal. If there are any other objects between the anchor and target nodes, you will have to move the cursor around them.

6.5 Deleting links from the Hyper-plate Editor

When a node is deleted all horizontal and vertical arrows associated with it are also erased automatically from the desktop, and their link information deleted from the hyper-plate record. However, the system cannot erase arrows that go around objects on the desktop. Note that the link information for these arrows, also, is deleted from the hyper-plate record; only the arrows are not erased from the desktop. These arrows will have to be erased using the <BACKSPACE> key. Links can be deleted manually also. Select "Del Link" from the menu. The system prompts you to place the cursor at any one end of the arrow and press the <ENTER> key. If you place the cursor at an invalid position on the desktop, the system warns you. You then move the cursor to the other end of the arrow and press the <ENTER> key again. The operation cannot be canceled once the deletion of the arrow has commenced.

6.6 Renaming a Node

Select the option "Rename" from the Hyper-plate editor menu and press the <ENTER> key. The system will prompt you for the name of the node to be renamed, and then, the new name.

If you enter an invalid name an error message is displayed. The new name is then displayed inside the rectangle representing the node.

6.7 Saving a Hyper-plate

Select the option "Save" from the Hyper-plate editor menu and press the <ENTER> key. The system displays the current file name of the hyper-plate and prompts you for a new name. You can either save the hyper-plate under the current name or type in a different name. The operation can be canceled by pressing the <ENTER> key.

6.8 Exiting the Hyper-plate

Select "Quit" and press the <ENTER> key. The system asks, whether you want to save the hyper-plate. If you enter "Y," which is "yes," it prompts you for a name. If a file of the same name exists, the system asks you whether to overwrite the current hyper-plate file. The operation can be canceled by pressing the <ESC> key.

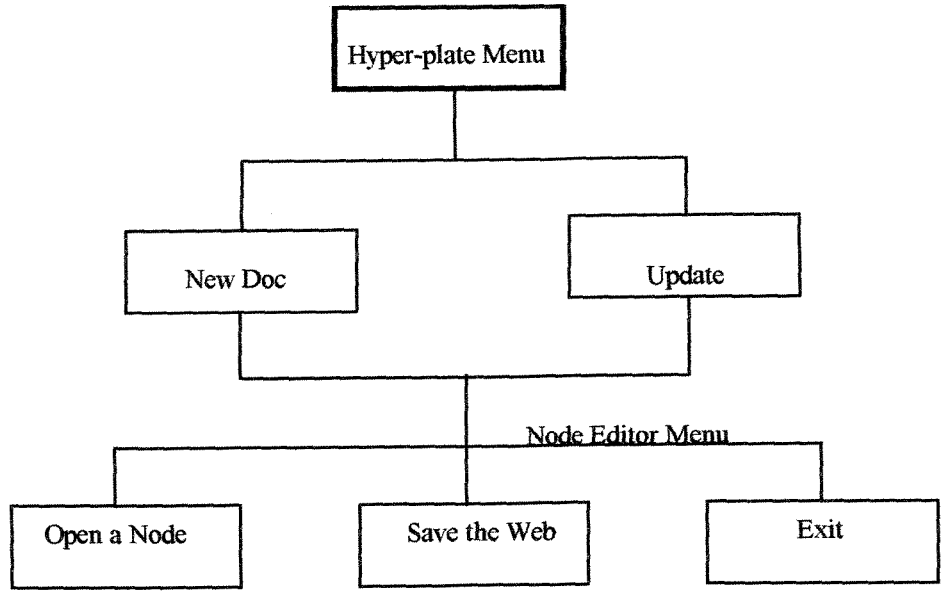

Figure A.3. Accessing the Node Editor from the Hyper-plate Menu.

7. Node Editor and Menu

Node editor menu (Figure A.3) offers three selections to the user: "Open a Node," "Save the Web," and "Exit." You can access the Node editor and Node menu from the Hyper-plate menu by either selecting "New Doc" or "Update." When you select "New Doc," a new hypertext document is created. The system assigns a file name for the new document. It is a "W" with the hyper-plate name appended to it. Note that all hypertext document (web) files have ".WEB" extension in their file name.

If you select "Update," the system asks you for the name of the hypertext document to be updated. You can type in a name. Alternatively, you can get a list of web files by pressing the <ENTER> key, and select a file from the displayed list. You will be then prompted to type in

the name of the hyper-plate to be used to update this web. You can type in the name or get a list of the hyper-plate files by pressing the <ENTER> key.

7.1 Opening a Node

Select "Open a Node" from the Node editor menu and press the <ENTER> key. Position the cursor inside the node to be accessed and press the <ENTER> key again. If you place the cursor at a location on the desktop where no node exists, an error message is displayed. An edited node will be marked by a rectangle of different color.

7.2 Saving the web

Select the option "Save the Web" and press the<ENTER> key. You will not be prompted for a name. If the web is a new web, the system automatically assigns a file name. If you are updating a web, the file will be saved under its existing file name.

7.3 Exiting the Node Editor

Select the option "Exit" and press the <ENTER> key. The system automatically saves the web for you. You will not be prompted for a name.

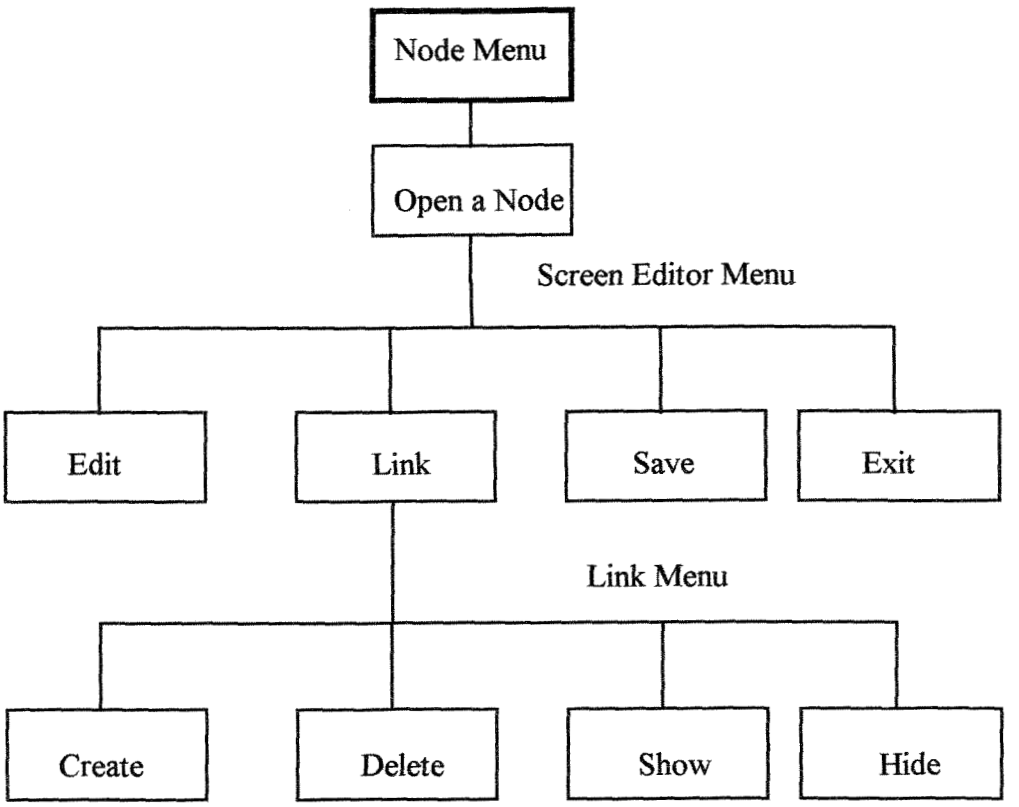

Figure A.4. Hierarchy of Node Menu.

8. Screen Editor and Menu

Screen editor and Screen menu (Figure A.4) are accessed from the Node Menu by selecting "Open a Node" and pressing the <ENTER> key. The system starts the screen editor. It then loads the screen of the node to be edited in the screen editor. The name of the screen you are editing is displayed at the top, and the buttons (links) at the bottom. You can change the name of the screen.

8.1 Editing a Screen

Select "Edit" and press the <ENTER> key. The cursor will move into the desktop area. Pressing the <ESC> key will enable you to make another menu selection. In the screen editor, you can type in text, create rectangles, draw lines, move a block of text, and change the color of a section of text. For the commands to do these editing tasks, press the help key F1.

8.2 Saving a screen

Select the menu option "Save" and press the <ENTER> key. The system will save the contents of the screen. You will not be prompted for a file name. All screen files have the extension ".SCR" in their file name.

8.3 Exiting the Screen Editor

Select the menu option "Exit" and press the <ENTER> key. When you exit, the screen will be automatically saved for you. You will not be prompted for a file name.

9. Link Menu

Selecting "Link" in the screen editor will display the Link menu (see Figure A.4). You can create more links, or delete the links. You can also delete the links created by the Hyper-plate. However, we caution you not to delete the links created by the Hyper-plate unless it is absolutely necessary. Deleting these links may introduce inconsistencies in the document structure. From the Link menu you can go back to the Screen editor by pressing the <ESC> key.

9.1 Creating a Link

Select the option "Create" from the Link menu and press the <ENTER> key. You will be prompted to highlight the text that you want to use as the button. You will be then prompted to type the name of the target node to link this button. You can type in a name or get a list of the node (screen) files that are in the system. Highlight the node name and press the <ENTER> key.

9.2 Deleting a Link

Select the "Delete Link" option from the Link menu and press the <ENTER> key. Highlight the link to be deleted and press the <ENTER> key again.

Appendix B

Hyper-plate Detailed Design and Implementation

Hyper-plate Detailed Design and Implementation

In the design and implementation of Hyper-plate, we have adhered to structured design and implementation techniques. Modules have been designed to group related functions. The functions operate by parameter passing. You may notice some functions use global variables. This was necessitated because of the original design of HypAS. The Hyper-plate modules are included in the HypAS project file and compiled into one executable file, HYPAS.EXE. An installation utility, INSTALL.EXE, is created to move HypAS from floppy to hard disk and configure the system. In this appendix we describe the modules developed for Hyper-plate, and the functions implemented in each module.

HYPAS PROJECT

The modules developed for the Hyper-plate system are combined with the HypAS modules to create a single project file: HYPAS.PRJ. The modules of Hyper-plate system are indicated with an asterisk. For the detailed design of HypAS modules reader is referred to the working paper of Abdalla [ABD93b].

Project file: HYPAS.PRJ

*ARROWS.CPP
BLOCK_F.CPP
BLOCK_M.CPP
CONFIG_F.CPP
CONFIG_M.CPP
EDHELP_F.CPP
EDIT_F.CPP
EDIT_M.CPP
HYPAS_M.CPP
LINEDR_F.CPP
LINEDR_M.CPP
LINK_F.CPP
LINK_M.CPP
*LINK_OBJ.CPP
LOAD_F.CPP
LOAD_M.CPP
*NODE_M.CPP
*SCR_EDIT.CPP
SAVE_F.CPP
SAVE_M.CPP
*TBLOCK_F.CPP
*TEDIT_F.CPP
*TEDIT_M.CPP
*TEM_F.CPP
*NODE_OBJ.CPP
*TLINE_F.CPP
*TLINK_M.CPP
*TLINK_F.CPP
*TWEB_F.CPP
VIEW_F.CPP
WEB_F.CPP
WEB_M.CPP
TOOLKIT.CPP

Module: ARROWS.CPP

This module creates and deletes links automatically on the Hyper-plate desktop. It also contains the functions that create and delete manual links. These functions are invoked by the Hyper-plate editor.

Functions:

boolean auto_arrow
(T_type* template, char* anchor_name, char* target_name, int arrow_type)
void DrawArrow(int arrow_type, int* x_position, int* y_position)
void EraseArrow(int* x_position, int* y_position)

Module: NODE_M.CPP

This module defines the functions for displaying the Node menu and the related functions. They are invoked from the Hyper-plate menu.

Functions:

void obj_open_menu(T_type* template)
void open_menu(int menu_choice)

Module: SCR_EDIT.CPP

This module defines the screen menu. Screen menu is invoked by the node menu when we open a node for editing.

Functions:

int scr_menu(int* x_position, int* y_position, char* node_name)
void display_menu(int menu_selection)

Module: TBLOCK_F.CPP

This module contains functions that create double bordered and single bordered rectangles. In addition, it also defines functions for copying, moving, erasing and coloring blocks of text. These functions are invoked from the screen editor. The hyper-plate editor can invoke only the functions to create double bordered and single bordered rectangles.

Functions:

select_type tselect_block
(int x_pos, int y_pos, int bMINX, int bMINY, int bMAXX, int bMAXY)
void single_border
(int FIRSTX, int FIRSTY, int SECONDX, int SECONDY, char* fname, int color)
void double_border
(int FIRSTX, int FIRSTY, int SECONDX, int SEONDY, char* fname, int color)
void copy_block
(int FIRSTX, int FIRSTY, int SECONDX, int SECONDY, int x_pos, int y_pos)
void erase_block(int FIRSTX, int FIRSTY, int SECONDX, int SECONDY)
void move_block
(int FIRSTX, int FIRSTY, int SECONDX, int SECONDY, int x_pos, int y_pos)
void color_block(int FIRSTX, int FIRSTY, int SECONDX, int SECONDY, int color)
void cancel _block(void)

Module: TEDIT_F.CPP

This module defines the functions of Hyper-plate editor, Hyper-plate edit menu, and related functions. It provides the desktop where a Hyper-plate editor creates rectangles to represent nodes and arrows to represent links.

Functions:

void temp_editor(T_type* template)

void editor_info(int x_position, int y_position)

int menu(int* x_pos, int* y_pos, T_type* template)

void display_menu(int menu_selection)

Module: TEDIT_M.CPP

This module contains the functions for displaying the hyper-plate menu. They are invoked from the main menu when the user selects Hyper-plate option.

Functions:

int edit_menu(void)

void edit_menu(int menu_choice)

Module: TEM_F.CPP

This module defines the functions for creating, initializing, opening, saving, updating, and viewing a hyper-plate. Some of the functions are invoked from the Hyper-plate menu, and other from the Hyper-plate edit menu.

Functions:

boolean create_temp(T_type* template)

boolean open_temp(T_type* template)

boolean save_temp(T-type* template)

void init_temp(T_type* template)

void update_temp(T_type* template)

void view_temp(void)

Module: NODE_OBJ.CPP

This module defines the functions for creating, initializing, deleting, accessing and saving a node. Most of the functions are invoked from the Hyper-plate edit menu. Some functions are invoked from the node edit menu.

Functions:

boolean create_obj(T_type* template, int x_pos, int y_pos)

boolean delete_obj(T_type* template, char * fname, int * x_pos, int* y_pos)

boolean access_obj(T_type* template, char * fname)

void node_compress(int ndx, T_type* template)

boolean search_temp(T_type* template, char* node_name)

boolean search_node(int x_pos, int y_pos, obj_type node)

int move_cursor(int* x_pos, int* y_pos)

void clear_node(int FIRSTX, int FIRSTY, int SECONDX, int SECONDY)

boolean save_obj(char* object_name)

boolean get_obj(obj_type* node, int* x_pos, int* y_pos)

int read_str(char* fname)

Module: LINK_OBJ.CPP

This module defines the functions for creating links, deleting links, and automatically erasing the arrows from the desktop. They are invoked from the Hyper-plate editor menu.

Functions:

void create_link(T_type* template)

void delete_links(T_type* template)

void erase_link(T_type* template, int* x_pos, int* y_pos)

void link_compress(int ndx, T_type* template)

boolean search_link(link_type* link, int x_pos, int y_pos)

boolean clear_link(link_type link)

Module: TLINK_M.CPP

This module defines the functions for the Link menu. Link menu is accessed from the screen menu.

Functions:

boolean link_menu(T_type* template)

void display_link_menu(int menu_choice)

Module: TLINK_F.CPP

This module contains the definition of functions for creating and deleting the link buttons. These functions are invoked from the link menu.

Functions:

boolean create_link(T_type* template)

boolean delete_link(T_type* template)

void show_link(void)

void hide_link(void)

char get_target_type(T_type* template)

Module: TWEB_F.CPP

This module defines the functions for creating, initializing, saving, and configuring a hypertext web. They are invoked by the hyper-engine.

Functions:

void new_web(char* fname)

void make_web(T_type* template)

void init_web(char* fname)

void init_config(char* fname)

void load_config(char* fname)

void save_config(char* fname)

void init_web_fields(void)

void init_web_list(void)

void load_web(char* fname)

void save_web(char* fname)

void update_web(char* fname)

int find_web_slot(char* fname)

void save_web_record(char* fname)

int look_up_record(char* node_name)

void init_web_record(char* fname)

int find_web_record(char* fname)

void load_web_record(char* fname)

int add_web_record(char* fname)

int delete_web_record(char* fname)

int insert_web_record(char* fname)

void modify_record(char* fname)