State of the Art Review on Hypermedia Issues And Applications

V. Balasubramanian
Graduate School of Management, Rutgers University, Newark, New Jersey
bala@pegasus.rutgers.edu

Converted to HTML by Denys Duchier, March 1994.
duchier@cs.sfu.ca

[*Index*] [*Next*] [*Postscript*]

CHAPTER 1

HYPERTEXT - AN INTRODUCTION

1. Introduction

Hypertext systems are emerging as a new class of complex information management systems. These systems allow people to create, annotate, link together, and share information from a variety of media such as text, graphics, audio, video, animation, and programs. Hypertext systems provide a non-sequential and entirely new method of accessing information unlike traditional information systems which are primarily sequential in nature. They provide flexible access to information by incorporating the notions of navigation, annotation, and tailored presentation [Bieber, 1993]. There are a number of research issues related to the design, development, and application of hypertext systems. This paper is a review of literature related to all these issues. This chapter is an introduction to hypertext, some existing systems, and some pioneers who have contributed to the definition and understanding of many aspects related to hypertext. Chapter 2 discusses issues related to hypertext implementation. Chapter 3 is on database requirements for hypertext systems. Chapter 4 discusses user interface issues and evaluation of hypertext. Chapter 5 is on information retrieval in hypertext systems. Chapter 6 discusses research efforts in the area of integrating hypertext with the work environment. Chapter 7 discusses some of the applications for which the hypertext paradigm is most suitable. Chapter 8 discusses a systematic approach to user interface design for a hyprtext system. It is an attempt to apply some of the ideas discussed in earlier chapters. Chapter 9 is a summary of all research issues and sets some directions for further work.

1.1 Hypertext

Hypertext has been defined as "an approach to information management in which data is stored in a network of nodes connected by links (Figure 1). Nodes can contain text, graphics, audio, video as well as source code or other forms of data." [Smith & Weiss, 1988]. Hypertext with multimedia is called "hypermedia". The promise of hypermedia[1] lies in its ability to produce large, complex, richly connected, and cross-referenced bodies of information.

In 1965, Nelson coined the word "hypertext" (non-linear text) and defined it as "a body of written or pictorial material interconnected in a complex way that it could not be conveniently represented on paper. It may contain summaries or maps of its contents and their interrelations; it may contain annotations, additions and footnotes from scholars who have examined it." [Nelson, 1965].

The original idea of hypertext was first put forth by Bush in July 1945. He described a device called "memex" in which an "individual stores his books, records and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory." [Bush, 1945]. He described the essential feature of memex as its ability to tie two items together.

The essential feature of hypertext, as defined in recent years, is the concept of machine-supported links (both within and between documents). It is this linking capability which allows a nonlinear organization of text.

Outside the academic world, due to the implementation of hypertext-like features in products such as MS Windows Help, information systems professionals are of the opinion that hypertext is just another user interface approach. However, hypertext is a hybrid that spans across traditional boundaries. It is a database method providing a novel way of directly accessing and managing data. It is also a representation scheme, a kind of semantic network, which mixes informal textual material with more formal and mechanized processes. It is an interface modality that features link icons or markers that can be arbitrarily embedded with the contents and can be used for navigational purposes [Conklin, 1987]. In short, a hypertext system is a database system which provides a totally different and unique method of accessing information. Whereas traditional databases have some structure around them, a hypertext database has no regular structure [Nielsen, 1990]. The user is free to explore and assimilate information in different ways.

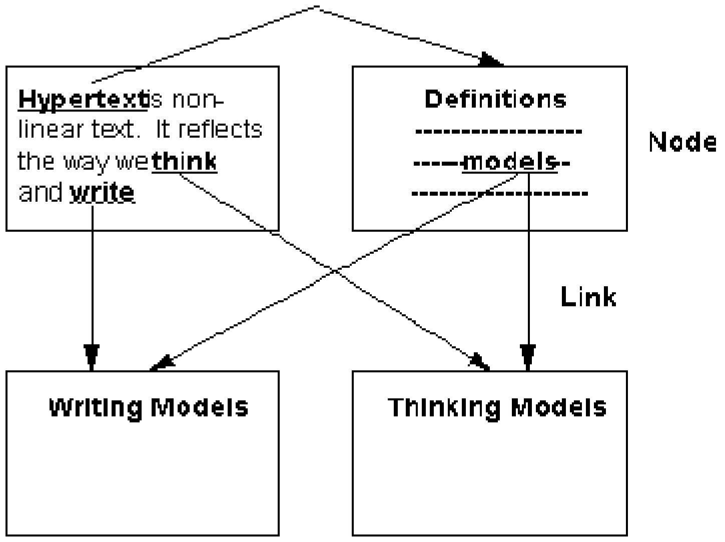
Hypertext is non-linear text. It reflects the way we think and write
Definitions
models
Node
Link
Writing Models
Thinking Models

Figure 1.1 Example of a Hypertext Document

1.2. Nodes and Links

A hypertext system is made of nodes (concepts) and links (relationships). A node usually represents a single concept or idea. It can contain text, graphics, animation, audio, video, images or programs. It can be typed (such as detail, proposition, collection, summary, observation, issue) thereby carrying semantic information [Rao & Turoff, 1990]. Nodes are connected to other nodes by links. The node from which a link originates is called the reference and the node at which a link ends is called the referent. They are also referred to as anchors. The contents of a node are displayed by activating links.

Links connect related concepts or nodes. They can be bidirectional thus facilitating backward traversals. Links can also be typed (such as specification link, elaboration link, membership link, opposition link and others) specifying the nature of relationship [Rao & Turoff, 1990]. Links can be either referential (for cross-referencing purposes) or hierarchical (showing parent-child relationships). Activation of link markers display nodes.

2. Basic Features of a Hypertext System

1. A Graphical User Interface, with the help of browsers and overview diagrams, helps the user to navigate through large amounts of information by activating links and reading the contents of nodes.

2. An authoring system with tools to create and manage nodes (of multiple media) and links.

3. Traditional information retrieval (IR) mechanisms such as keyword searches, author searches etc. There are also attempts to incorporate structure queries along with content queries - retrieving a part of the hypertext network based on some user-specified criteria.

4. A hypermedia engine to manage information about nodes and links.

5. A storage system which can be a filesystem or a knowledge base or a relational database management system or an object-oriented database management system.

3. Systems and People

The concept of hypertext has been around for a long time. The dictionary and the encyclopedia are very old forms of hypertext. These can be viewed as a network of textual nodes joined by referential links. The Talmud, with its heavy use of annotations and nested commentary, and Indian epics such as Ramayana and Mahabharata (stories branching off to other stories) are ancient prototypes of hypertext representation. We will review some of the well-known systems which have been implemented and widely studied by researchers in understanding various issues related to hypertext.

3.1 Memex

Bush is considered the "grandfather" of hypertext. He proposed a system called the "memex" as long ago as 1945 [Bush, 1945]. Though the system was never implemented, the concepts are still relevant to this day. Bush was concerned about the explosion of scientific literature which made it impossible even for specialists to follow developments in a field. He felt the need for a system that would help people find information more easily than was possible on paper.

The Memex would store information on microfilm which would be kept on the user’s desk. The desk would contain many translucent screens on which several microfilms could be projected for convenient reading. The would also be a keyboard and sets of buttons and levers.

The Memex would have a scanner for user input of new material and it would also allow users to make handwritten marginal notes and comments. Apart from the conventional form of indexing, Bush proposed "associative indexing, the basic idea of which is a provision whereby any item may be caused at will to select immediately and automatically another. This is the essential feature of memex. The process of tying two items together is the important thing." [Bush, 1945].

3.2 Augment/NLS

As part of the Augment Project, primarily designed for office automation, Engelbart of SRI developed a system called NLS (oN Line System) which had hypertext-like features. This system was used to store all research papers, memos, and reports in a shared workspace that could be cross-referenced with each other [Engelbart, 1963]. In 1968, he demonstrated NLS as a collaborative system among people spread geographically.

3.3 Xanadu

For thirty two years now, Nelson has been working on his vision of a "docuverse" (document universe) where "everything should be available to everyone. Any user should be able to follow origins and links of material across boundaries of documents, servers, networks, and individual implementations. There should be a unified environment available to everyone providing access to this whole space." [Nelson, 1987].

Nelson designed Xanadu, a repository publishing system "intended to store a body of writings as an interconnected whole, with linkages, and to provide instantaneous access to any writings within that body." [Nelson, 1980]. This system has no concept of deletion. That is, it is a write-once system. Once something is published, it is for the entire world to see forever. As links are created by users, the original document remains the same except for the fact that a newer version is created which would have references to the original version(s). Since conventional file systems are not adequate to implement such a system, Project Xanadu has focused much of its attention on the re-design and re-implementation of file systems. This, in turn, required the creation of a whole new operating system incorporating a hypertext engine. The back-end for the system was scheduled to be released on Sun Workstations during 1992.

3.4 Intermedia

The Intermedia system, developed at Brown University’s Institute for Research and Information Scholarship, is an integrated environment that allows different types of applications (word processors, editors, and other programs) to be linked together. It is a collection of tools that allows authors to link together the contents of text, timeline, graphics, 3-D models and video documents over a network of high-powered workstations [Meyrowitz, 1986]. The applications that exist within the Intermedia framework include a text editor (InterText), a graphics editor (InterDraw), a scanned image viewer (InterPix), a three-dimensional object viewer (InterSpect), and a timeline editor (InterVal).

The hypermedia functionality of the system is integrated into each application so that the creation and traversal of links can be intermixed with the creation and editing of documents. The system provides consistent, modeless, direct-manipulation applications. Strict conformance to user interface standards throughout the system makes it easy for the user to interact with all the applications in a similar manner.

Intermedia supports the concept of webs, composite entities that have many nodes and links between them. A link can belong to one or more webs. It provides three types of navigation tools: paths, maps, and scope lines. It supports shared and concurrent access to documents based on a system of access permissions. Intermedia has been used in presenting two courses online - English literature and biology [Yankelovich et al., 1988].

3.5 NoteCards

NoteCards is a hypermedia system for designers, authors, and researchers to analyze information, construct models, formulate arguments, and process ideas [Halasz, 1988]. Its basic framework is a semantic network composed of notecards connected by typed links. It provides users with tools for displaying, modifying, manipulating, and navigating through the network.

NoteCards contains four basic constructs: notecards, links, browsers, and fileboxes. Notecards contain information embedded in text, graphics, images, voice or other media. Links represent binary relationships between cards. Browsers display node-link diagrams of portions of the network. Fileboxes provide a mechanism to organize cards into topics or categories. NoteCards can be integrated with other systems running in the Lisp environment such as mail systems, databases, and expert systems.

3.6 KMS

Knowledge Management System (KMS), a descendant of ZOG, was developed at Carnegie Mellon University. It was designed to manage fairly large hypertext networks across local area networks. KMS is based on the basic unit called the frame. A frame can contain text, graphics, or images. Frames are connected to other frames via links. Links are of two types: tree items to represent hierarchical relationships and annotation items to represent referential relationships. In KMS, there is no distinction between browsing and authoring modes. Users can make changes to a frame or create links at any time and these changes are saved automatically [Acksyn et al., 1988].

KMS supports features such as aggregation, keyword searching, tailorability, collaboration, concurrency control, data integrity and security. It has been used for collaborative work, electronic publishing, project management, technical manuals and electronic mail.

3.7 HyperTies

HyperTies started as TIES (The Interactive Encyclopedia System) under the direction of Shneiderman at the University of Maryland’s Human-Computer Interaction Laboratory. It provides authoring and browsing tools. A node may contain an entire article that may consist of several pages. Links are represented by highlighted words or embedded menus which can be activated using the keyboard or a touchscreen. Readers can preview links before actually traversing them. The user interface is relatively simple due to the original emphasis on museum information systems or kiosks. The commercial version is being used for a much wider spectrum of applications such as diagnostic problem solving, self-help manuals, browsers for libraries, and on-line help [Shneiderman, 1988].

3.8 Guide

Guide was developed by Peter Brown as a research project at the University of Canterbury, U.K. Later, it was commercially marketed by Office Workstations Limited both on the IBM PC and Apple

Macintosh. It is the most popular commercial hypertext system. Text and graphics are integrated together in articles or documents. Guide supports four different kinds of links: replacement buttons, note buttons, reference buttons, and command buttons. Navigation through the replacement buttons initially provides a summary of the information and the degree of detail can be changed by the reader. Similar to KMS, Guide also does not distinguish between the author and the reader [Brown, 1987].

3.9 Textnet

Textnet was designed and implemented by Trigg at the University of Maryland. It was developed to support the on-line scientific community in text creation, footnoting, annotating and critiquing. Textnet is a hypertext system based on a semantic network of nodes and labeled links. Nodes can be either primitive pieces of text called chunks or composite hierarchies called table of contents (tocs). There are two basic types of links: normal links and commentary links. In addition, there are about eighty different types of links with different functions [Trigg &Weiser, 1986].

3.10 Writing Environment (WE)

Researchers at the University of North Carolina at Chapel Hill developed the Writing Environment (WE), a hypertext system based on a cognitive model of the communication process [Smith et al., 1987]. This model explains reading as the process of taking linear streams of text, comprehending it by structuring the concept hierarchically, and absorbing into long-term memory as a network. Writing is seen as a reverse process: A loosely structured network of internal ideas and external sources is first organized into an appropriate hierarchy or outline which is then translated into a linear stream of words, sentences, paragraphs, sections, and chapters [Smith et al., 1987].

WE was designed to support the process of writing. It contains two major view windows, one graphical and another hierarchical along with commands. The graphical windows allows the user to loosely structure their ideas in terms of nodes. As some conceptual structure begins to emerge, the writer can transfer the nodes into the hierarchy window which has specialized commands for tree operations. WE uses a relational database for the storage of nodes and links in the network. There are three other windows: an editor window, a query window, and a window to control system modes and the current working set of nodes. WE can be used both as a hypertext system as well as an authoring system with advanced graphical, direct manipulation structure editing capabilities [Conklin, 1987].

Other notable systems include: Hypertext Editing System (HES) and File Retrieval and Editing System (FRESS) from Brown University, MCC’s Group Issue Based Information System (gIBIS).

4. Reading and Writing Models

Hypertext parallels human cognition and facilitates exploration. We think in nonlinear chunks which we try to associate with each other and build a network of concepts. When we read a book, we go back and forth a number of times to refer to previously read material, to make notes, and to jump to topics using the table of contents or the index. When we set out to write a document we first develop an outline of ideas. Then, we brainstorm, write down on paper, organize, revise, reorganize and repeat the cycle till we are satisfied with the outcome - a coherent document. In fact, we have been forced to adapt to traditional, linear text because of representation on paper.

In order to understand hypertext, it is very essential to understand how people read and write documents. Reading and writing models have been developed by cognitive psychologists that can be used to understand non-linear thinking by human beings [Rada, 1991], [Smith et al., 1987].

4.1 Reading Model

The theory of semiotics or the study of symbols shows that the understanding of knowledge takes place at four levels : lexical, syntactic, semantic and pragmatic [Rettig, 1992], [Rada, 1991]. At the lexical level, the user determines the definition for each word encountered. At the syntactic level, the subject, action and object of a sentence are determined. The meaning of a sentence is determined at the semantic level. The pragmatic interpretation of text depends on the integration of semantic meaning of text with the reader's knowledge of self and of the world.

While reading text, people proceed from a lexical level to the syntactic level, to the semantic and to the pragmatic levels in that order. All these levels interact continuously and they cannot be truly separated. The reader might have to have knowledge of the world in order to understand the meaning of a word. The correct syntactic and semantic interpretation of text may depend on the reader's knowledge of the world. Hence, though readers may proceed from words to sentences, to paragraphs and to the overall document, the progress is more forward and backward.

A mental representation of the meaning of text is then constructed which is in the form of propositions or relationships. While reading text, readers establish local coherence in short-term memory - small scale inferences from few small units of information (relationships between words, sentences and so on) [Thuring et al., 1991]. The reader makes preliminary hypotheses based on titles, words, propositions, and knowledge about the real world. A reading control system retrieves knowledge from the real world, present in long-term memory, in order to filter out information present in short-term memory. These hypotheses are refined as the reading of the text proceeds with the reading control system being invoked continuously. These propositions are combined into larger structures, also called global coherence [Thuring et al., 1991]. This hypothesized macroproposition or superstructure is used to understand the overall content of the text. The construction of a coherent mental representation has important consequences for navigation. In addition to generating forward references, we accumulate cues for backward navigation.

The reading control system uses the spreading activation model to access propositions or concepts. In semantic memory, each concept is connected to a number of other concepts. Activating one concept activates its adjacent concepts which in turn activate their adjacent concepts. Thus, activation spreads through the memory structure, determining what is to be added and what is to be removed from the interpretation of text. This process continues until further activation of adjacent propositions does not change the propositions used to interpret the text. That is, spreading activation decreases over time and semantic distance.

4.2 Writing Model

Writing is constrained by goal and audience. The author is guided by a goal but constrained by what the audience is prepared to accept. Different people approach writing in different ways. Some are good at making an outline first and then brainstroming. Some do the opposite. An expert author would always keep the reading model in mind so that the writing clearly reaches the target audience.

Writing involves the following three phases : exploring, organizing, and encoding [Rada, 1991]. In the Cognitive Framework for Written Communication (Figure 2), Smith et al. call these three phases: prewriting, organizing, and writing [Smith et al., 1987].

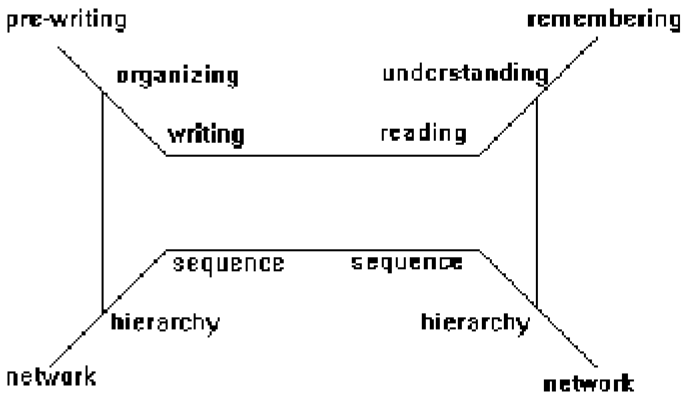

Figure 1.2 Cognitive Framework for Written Communication [Smith et al., 1987].

Exploring or pre-writing is the process of brainstorming and taking unstructured notes. The writer retrieves potential content from long-term memory or external sources, considers possible relations among ideas, groups related ideas and constructs small hierarchical structures. Thus, the product of exploration is a network or directed graph of ideas.

Organizing is the process of putting these notes or ideas in order, in the form of an outline or a hierarchy. This process involves abstract construction that involves perceiving subordinate/ superordinate relations, comparing abstractions, sequencing, proportion, and balance. Thus, the product of organization is a hierarchy of related concepts.

Encoding or writing is the final phase of completing the document. The primary task is translating the abstractions of content and the relations of a hierarchical structure into a sequence of words, sentences, paragraphs, sections, chapters, and illustrations. The structure of the encoded text is linear and represents a path through the hierarchy.

It is interesting to note that reading employs processes in the reverse order. That is, a linear sequence of words is transformed into a hierarchy which is later integrated into a network in long-term memory [Smith et al., 1987].

The writing model can be extended by considering unstructured and structured representation at each phase. Whereas an unstructured item is isolated, a structured item shows coherence. Exploring can be split into unstructured brainstorming followed by structured note-taking. Organizing can be classified as unstructured argumentation where relationships are established between ideas and structured organization of notes where notes are grouped together to make coherent sense. Encoding has an unstructured phase of linear planning which involves viewing groups of notes as sequences and a structured phase of drafting and revising in order to produce a final document (which is a linear sequence of notes).

Just as the reader of a linear document constructs a local and global mental representation of the document, the author of a linear document uses cues both at the local and at the global levels, dividing the document into chapters, sections, paragraphs, sentences, words etc. This facilitates comprehension and navigation [Thuring et al., 1991].

Thus, both reading and writing processes emphasize a lot on the non-linear nature of thinking, a natural process in human beings. Human cognition is essentially organized as a semantic network in which concepts are linked together by associations. Hypertext systems try to exploit this basic nature of cognition.

5. Summary

A hypertext is a database system which provides a unique and non-sequential method of accessing information. The essential features of hypertext are nodes and links. Nodes can contain text, graphics, audio, video, animation, and images while links connect nodes related in a certain manner. It is the linking capability which allows the non-linear organization of text. We have seen some of the pioneers in this field and some of the systems they built which have paved the way to understand better the theoretical and practical aspects of hypertext.

References

[Acksyn et al., 1988]. Acksyn, Robert M., McCracken, Douglas L., & Yoder, Elise, A. KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations, CACM, July 1988.

[Bieber, 1993]. Bieber, Michael. Providing Information Systems with Full Hypermedia Functionality, Proceedings of the Twenty-Sixth Hawaii International Conference on System Sciences, 1993.

[Brown, 1987]. Brown, Peter J. Turning Ideas into Products: The Guide System, Hypertext ’87 Proceedings, November 1987.

[Bush, 1945]. Bush, Vannevar. As We May Think, The Atlantic Monthly, July 1945.

[Conklin, 1987]. Conklin, Jeff. Hypertext: An Introduction and Survey, IEEE Computer, September 1987.

[Engelbart, 1963]. Engelbart, Douglas C. A Conceptual Framework for the Augmentation of Man’s Intellect, Vistas In Information Handling, Volume 1, Spartan Books, Washnigton D.C., 1963.

[Halasz, 1988]. Halasz, Frank G., NoteCards: A Multimedia Idea Processing Environment, Interactive Multimedia edited by Ambron, Sueann and Hooper, Kristina, Microsoft Press, 1988.

[Meyrowitz, 1986]. Meyrowitz, Norman K., Intermedia: The Architecture and Construction of an Object-Oriented Hypermedia System and Applications Framework, OOPSLA ’86 Proceedings.

[Nelson, 1965]. Nelson, Ted. A File Structure for the Complex, The Changing and The Indeterminate, ACM 20th National Conference, 1965.

[Nelson, 1980]. Nelson, Ted. Replacing the Printed Word: A Complete Literary System, Information Processing ’80, 1980.

[Nelson, 1987]. Nelson, Ted. All for One and One for All, Hypertext ’87 Proceedings, November 1987.

[Nielsen, 1990]. Nielsen, Jakob. Hypertext/Hypermedia. Academic Press, 1990.

[Rada, 1991]. Rada, Roy. Hypertext: From Text to Expertext, McGraw Hill Publishers, 1991.

[Rao & Turoff]. Rao, Usha & Turoff, Murray. Hypertext Functionality: A Theoretical Framework, International Journal of Human-Computer Interaction, 1990.

[Rettig, 1992].

[Shneiderman, 1988]. Shneiderman, Ben. User Interface Design for the HyperTies Electronic Encyclopedia, Hypertext ’87 Proceedings, November 1987.

[Smith & Weiss, 1988]. Smith, John & Weiss, Stephen. An Overview of Hypertext, CACM, July 1988.

[Smith et al., 1987]. Smith, John B., Weiss, Stephen F., and Ferguson, Gordon J. A Hypertext Writing Environment and Its Cognitive Basis, Proceedings of Hypertext ’87, ACM Press, 1987.

[Thuring et al., 1991]. Thuring, Manfred, Haake, Jorg M., and Hannemann, Jorg. Hypertext ’91 Proceedings, 1991.

[Trigg & Weiser, 1986]. Trigg, Randall & Weiser, M. TEXTNET: A Network Based Approach to Text Handling, ACM Transactions on Office Information Systems, January 1986.

[Yankelovich et al., 1988]. Yankelovich, Nicole, Haan, Bernard J., Meyrowitz, Norman K., Drucker, Steven M., Intermedia: The Concept and the Construction of a Seamless Information Environment, IEEE Computer, January 1988.

---

[1] In this paper, the terms hypertext and hypermedia are interchangeably used.

CHAPTER 2

IMPLEMENTATION ISSUES

1. Conversion of Text to Hypertext

In recent years, there has been a great amount of enthusiasm in converting every printed document into hypertext form. Though the rationale behind this is not always correct, manual hypertexts such as encyclopedias, dictionaries, training manuals are very well suited for hypertext conversion. These reference materials are not used the same way as other books. They are highly cross-referenced and are used in a non-linear fashion. Readers look for various structural cues such as table of contents, indexing by subject, keywords, authors, page numbers, sections, see-also listings etc.

1.1 Limitations of Printed Text

There are a lot of limitations imposed by the printed versions of reference books. These include [Cook, 1988]:

1. Even though some of the reference books run into many volumes, the amount of information that can be stored is still limited compared to electronic forms of storage. It is also difficult to search through large volumes of printed material.

2. They cannot be updated periodically.

3. Information search is predominantly lexical - the table of contents and the index provide the facility to jump to topics but the amount of cross-referencing is minimal. The printed index is limited by the size and selection criteria of the authors and does not always direct the user to all relevant information.

4. Information cannot be dynamically re-arranged to suit the individual needs of various kinds of users.

5. Information is spread over a number of volumes and after some time information retrieval becomes tedious.

1.2 Advantages of Hypertext Format

Enhancing encyclopedic information into hypertext format has many advantages [Raymond & Tompa, 1988], [Cook, 1988]:

1. Hypertext form can support good browsing capability.

2. Electronic media can store large amounts of information.

3. It can provide better visual prominence and more rapid navigation through huge number of entries - a

key mechanism to be employed in dynamic formatting of entries according to user specifications.

4. Most users refer to dictionaries, encyclopedias, and training manuals as part of more extended tasks. They would like to save their results and queries for future use, use annotation facilities, the ability to transfer OED text segments to other documents, tools to sort and filter quotations, and tools for statistical analysis of variables.

1.3 Conversion Issues

As much as they are well suited for adaptation to hypertext, converting text to hypertext has been a classic problem while dealing with very large information spaces such as training manuals, encyclopedias, dictionaries. Currently published literature on hypertext contains little work directly related to the scale of transforming large volumes of encyclopedic text into hypertext form (most deal with creating small hypertext documents, not converting large documents to hypertext).

The following are some of the issues involved in converting text to hypertext [Glushko, 1989], [Riner, 1991]:

1. Identifying documents that would benefit readers if converted to hypertext form.

2. Determining procedures to convert them to hypertext format.

3. Preparing documents in an electronic format from paper or other forms.

4. Identifying nodes and links and classifying them into various types (to capture semantics). An important problem related to this issue is called the fragmentation problem. It is still difficult to identify text units that can be separate modules and also serve as cross-references for other entries. Links should follow some model of the user's need for information in some particular context. Deciding on the level of granularity is a difficult problem. Too fine the granularity, greater the problem of fragmentation. Too coarse the granularity, greater the need or the display of large entries.

Also fragmentation tends to make an implicit structure (such as a subtle treatment of a theme that may communicate an idea more artistically) explicit, taking away the expressiveness of the statement. Therefore, we have to find means to reduce segmentation of ideas and loss of structural information due to the manipulation of the semantic structure of a linear document.

5. Determining the target of a link as a complete entry, a sub entry, or a derivative form is a challenging task. This involves determining the right part of speech, the etymological root, and applying sense-disambiguation to identify a particular meaning.

6. With present-day video monitors, the display of large entries in their entirety is still a problem. This can be partly solved by having fisheye views and abbreviations. Structural information can be extracted from the tags and employed in the construction of a structural view.

7. Performing the conversion and verifying the results.

2. Types of Conversion

There are two ways to convert existing documents into hypertext form - manual conversion and automated conversion [Riner, 1991].

2.1 Manual Conversion

Manual conversion involves using a hypertext authoring tool to create nodes and links manually. Hence, this process depends on the way the author (or the person who builds the hypertext form from a linear form) understands the structure and flow of the presented material. Being a repetitive process, it is prone to human error. There cannot be anything worse than a badly converted hypertext version of a linear document. Also, manual conversion is suitable only for small documents.

2.2 Automated Conversion

Automated conversion facilitates the easy identification of nodes and links based on pre-defined criteria. The output of an automated conversion process can be easily modified/enhanced by authors. Also, large information spaces such as dictionaries, encyclopedias, and training manuals can be converted to hypertext format very efficiently. As mentioned earlier, most linear documents have structural elements such as titles, sub-titles, chapters, sections, paragraphs, sentences, words, figures, tables, and indexes. An automated conversion system must be able to recognize these structural elements, identify nodes and links, and construct the appropriate links to form the hypertext network [Riner, 1991]. Links can capture both the hierarchical and referential nature of the material.

2.3 Guidelines for Conversion

The following can be some of the criteria in converting linear documents into hypertext format, both manually and automatically [Glushko, 1989]:

1. Utmost care is required while identifying text units as nodes that can be separate modules and still be sufficient enough to be cross-references for other entries.

2. A good design rule is to choose as the basic unit of text the smallest logical structure with a unique name (such as the title for an entry) - this can be used as a selection key in a hierarchical browser, in search lists as candidate keys, as bookmarks, and embedded cross-references.

3. Pages or paragraphs are less suited as hypertext units because they do not form convenient handles for manipulation.

4. It is very important to understand both the explicit and implicit link structures in the printed version of the material. Careful decisions have to be made as to what links to create and what to disregard.

5. It is important to understand the user’s task and to support links that follow some model of the user’s need for information in some particular context. It is essential NOT to link items that are related in idiosyncratic or superficial ways. Such hypertext links lead to "spaghetti documents". A careful analysis needs to be done as to what implicit and explicit hypertext structures users make use of in the linear

document.

6. The organization of the material should be open and flexible. Different kinds of views should be available for different users. For example, a repair manual can contain a training view, a troubleshooting view, a routine maintenance view and a purchaser’s view. View descriptions may appear as alternate overview diagrams or webs of information.

2.4 Automatic Link Construction

In most current systems, a large authoring effort is required to insert links into documents. Very little work has been done in the area of automatic link construction - links based on the semantic analysis of the underlying text. Such a feature requires considerable amount of analysis and the incorporation of an Artificial Intelligence engine.

In an effort towards automatic linking of hypertext nodes, Bernstein proposed a "link apprentice", a program that can examine a draft hypertext and create appropriate links. This can be done by establishing links based on the semantic analysis of the underlying text. Since these "clever" apprentices are intrinsically difficult to construct (they not only need precision but also accuracy and recall), he suggested a "shallow apprentice" - a system which discovers links through superficial textual analysis (of statistical and lexical properties) without analyzing meaning [Bernstein, 1990].

The shallow apprentice uses the Bloom filter method of text searching. Each hypertext page (node) has a Bloom filter hash table where each word is hashed. These hash tables are used to define a similarity between two hypertext pages by taking the normalized dot product of their hash tables. The apprentice will search the entire document checking for similarity to the page upon which the hypertext author is currently working. It will then retrieve the twenty pages that appear to be most similar to the current page.

The hypertext author can also construct hypertext paths or tours by choosing an interesting starting point and requesting the apprentice to construct a path through related material. However, the path may not be in logical order since the apprentice does not check for semantics.

3. Hypertext Templates

Hypermedia templates are defined as sets of pre-linked documents that can be duplicated [Catlin et al., 1991]. Another definition of a hypertext template states that it "is a partially-created, properly formatted collection of document skeletons that can be filled in by the user" [Rao & Turoff, 1990].

Templates automate the process of creating hypermedia collections by creating the "skeletons" of documents and linking them. They facilitate the design, organization, and presentation of a collection of knowledge in the form of hypertext.

The template can be considered as a composite object comprised of other objects such as nodes and links. The usage of a template will definitely speed up the process of an average user’s understanding of the underlying hypertext model or the metaphor. Without a template, a hypertext author will have to start constructing the hypertext collection of ideas from the beginning. Many applications such as collaborative writing, teaching aids etc., have some common basis that can be transformed into a hypertext template.

The following are some requirements for a hypertext system to provide templates:

1. It should provide some generic operations to create, duplicate, edit or delete a template. Duplication should yield empty documents with nodes and links.

2. There should be facilities to add contents to empty documents, list templates and their constituent documents and links, to display an overview of the template, to access a template by its type ("get a copy of the planning template"), by author, or by creation date.

3. There should be control operations to displaying an overview of the template, to zoom into specific link sets or webs or sub graphs and look at the contents of documents.

4. Strategic choices must exist to find out the master template from which a duplicate was created and to edit the master template. Editing a master template should propagate the changes to all templates created from it.

5. Facilities should exist to specify formats and screen layouts for a template and to add help.

6. Reactive choices must be provided to directly manipulate the contents of documents within a template such as editing, deleting, creating new links etc.

Intermedia, developed at Brown University, provides the following features for hypermedia templates [Catlin et al., 1991]:

1. Intermedia system provides the facility to create templates including the documents and links that make up the template. That is, a hypermedia author has the ability to create nodes, links, and link sets or webs within a template. A list of webs can be associated with a template one of which can be chosen as the default when the template is duplicated.

2. Documents within the same template can be linked. Users can also link a document in a template to another document outside of the template.

3. The user can specify the folder or directory under which each document is created and also the folder where the template has to be duplicated. The user can also name and save a template for future use. The system will make copies of all folders and documents and automatically link them just as the original template was linked. All new documents will be displayed for the purpose of editing.

4. When a template is duplicated, all associated documents and links can be easily accessed in new folders. The user will be prompted to choose one of the webs associated with the template. The user can open a template, add document members, delete members, rename them, create or modify links etc. Contents of documents can be edited.

5. The user can easily find out which template was used to make a new hypermedia collection.

6. The original template itself is write-protected so that users do not edit it accidentally.

Researchers at Brown University believe that the ability to duplicate collections of linked material can be extended to other hypertext environments. Research is required in the area of propagating editing changes to documents that were created using a particular template. The concept of class-based templates needs exploration - templates should be able to inherit characteristics from other templates (similar to the concept of inheritance in object-oriented systems). With inheritance, when an author changes a parent template, all of its sub-classed templates would change accordingly.

4. General Guidelines for Authoring Hypertext Documents

The comprehension and navigation of a hypertext document depends on the reader’s ability to construct a coherent mental representation. It is the author’s responsibility to ensure the construction of the hypertext document as a coherent entity. The construction of a coherent hypertext document can be considered to be a design problem. There are no established guidelines for writing hypertext documents. Guidelines have been developed, by Thuring et al., for the construction of a coherent hypertext document. Such a document should consist of the following three components - the content part, the organizational part, and the presentation part [Thuring et al,1991].

4.1 The Content Part

Nodes and links can be considered as design objects. Properties (semantics) can be associated with these design objects in order to introduce coherence in a hypertext document.

The content part contains design objects that carry information. They are content nodes that contain information and content links that connect content nodes based on semantic relationships. Content nodes can be either atomic or composite in nature. Content links can be typed specifying the exact nature of the semantic relationship. They can be classified into three types:

Level One: Links with no labels.

Level Two: Links with labels describing global semantic relationships such as "is discussed by", "is illustrated by".

Level Three: Links with more specific labels such as "is criticized by", "is shown graphically".

This classification of links is similar to the more elaborate classification of nodes and links in the hypertext framework developed by Rao and Turoff. According to Thuring et al., the author creating a hypertext document can initially create a Level Two link to show a general relationship between content nodes. As the author becomes more clear about the relationship between two nodes, the link labels can be changed to Level Three. Thus, "the levels of link label hierarchy support a continuous refinement of the links depending on author’s current state of knowledge." [Thuring et al., 1991].

While creating the content part, the following design rules can be applied:

a. Composite content nodes should be used to hierarchically structure the content of the document into domain specific sub-units of information.

b. The label of a link should be as specific as possible and should constitute a comprehensible sentence together with the names of the source and destination nodes.

4.2 The Organizational Part

Design objects of the organizational part increase coherence by structuring the network under a reader-oriented perspective. Using such an approach, the author can tailor variants of a document for different audiences.

Structure nodes organize content nodes and links in a specific manner. Each structure node has a name and a starting node. These can be of two types:

a. Sequencing nodes that allow the author to define the reading sequence through the content net. Readers can read only those content nodes that are determined by the sequencing node.

b. Exploration nodes allow the reader to explore - the reader can simply follow the content links to explore the subnet.

While sequencing nodes constrain the reader’s navigation through the document, exploration nodes allow unconstrained access to its content part.

Structure nodes can be connected by structure links which are also classified into two types:

a. Sequencing links associate the content of each sequencing node with a presentation sequence. They can be used to define ordering such as linear sequence, branching sequence etc.

b. Exploration links provide access to exploration nodes. An exploration link is embedded into a sequencing node and points to the beginning of an exploration node.

Sequencing nodes along with sequencing links can present different presentation sequences such as sequential paths, branching paths, and conditional paths.

The following design rules can be applied while creating the organizational part:

1. Choose an appropriate starting point to serve as an introduction to the document.

2. Construct appropriate paths based on reader’s interests and knowledge. This can be done by ordering sequencing nodes and links and providing additional information using exploration nodes and links. Thus, the author can create multiple versions of the document some having strictly linear sequences, some having branches, and conditional paths, and some a combination of all three.

Based on the above, the following hierarchy of design objects is derived.

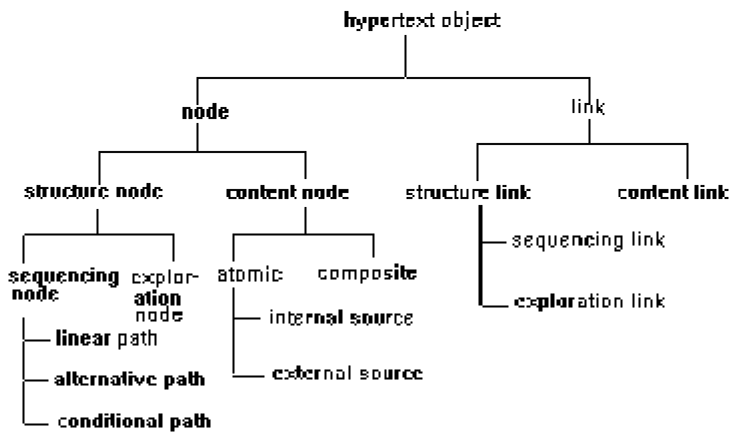

Figure 2.1 Hierarchy of Design Object Classes [Thuring et al., 1991].

4.3 The Presentation Part

The presentation part is concerned about the actual display of structure and content and provide the means of navigation. Authors can adopt three styles:

a. Textual Style: There is no graphical display of the structure, the presentation being limited to the display of the content of one or more nodes.

b. Graphical Style: There is a graphical display, such as an overview map, of the structure.

c. Combined Style: Both overviews and the ability to open nodes are provided.

The combination of the content part with the organizational part and the presentation styles would greatly facilitate comprehension and navigation.

5. Dynamic Hypertext

Halasz had identified dynamic or virtual structures, computation, and extensibility/tailorability as some of the issues to be addressed by next generation hypertext systems [Halasz, 1988]. Most current generation hypertext systems implement a static and explicit model of hypertext - nodes, links, and link markers must be declared explicitly and be fully enumerated during creation time as opposed to being declared dynamically and generated upon demand [Bieber, 1993].

Information systems such as Decision Support Systems (DSS) and Expert Systems require a dynamic implementation of hypertext, one that relies primarily on virtual structures and computation in order to generate a hypertext network in real time [Bieber, 1991]. Bieber developed a DSS shell that supported multiple DSS applications through a hypertext user interface. The user interface containing the hypertext engine provided DSS applications with hypertext functionality such as navigation, virtual structures, computation, and tailored presentation. Since many of the components making up the DSS are generated in real time as a result of user interaction, it was difficult to pre-define all nodes, links, and link markers at creation time.

Bridge laws were developed to determine the appropriate links and link markers automatically and embed them in the interactive application [Bieber, 1991]. These link markers provided access to reports, operations (DSS commands), and other components of DSS applications. Bridge laws are translation routines provided by the application to the hypertext interface. They map the elements defined in the application's original non-hypertext data or knowledge base to entities in the hypertext engine. They do not alter the application's data or knowledge bases.

A user interface control subsystem was developed to maintain global information about user profile, user-defined links, comments, application keywords and application supplied bridge laws. It was responsible for interpreting contents coming from the application (using bridge laws) to provide virtual link markers by highlighting the objects. It tailored different views of the application based on different sets of filters.

Stotts and Furuta describe virtual structures as dynamic adaptation of hypertext structure [Stotts & Furuta, 1991]. It involves collecting information from user interaction with a hypertext system, making inferences and decisions based on this information and creating appropriate physical changes in the document at appropriate times. Adaptation can occur at two levels - behavior of the document (timing of sequences, providing automated help, presenting collections in parallel or in sequence etc) and structure of the document (the way the nodes are linked).

According to Stotts and Furuta, a hypertext document can be considered to have two layers - a fixed underlying information structure that is created by the hypertext author and a flexible structure that is superimposed on the former and is tuned to each user's requirements. The flexible layer can be generated dynamically. The manner in which information is organized and presented can be altered without actually changing the information relation contained in the original links. This is similar to Bieber's concept of bridge laws which simply map an application's non-hypertext data to a hypertext interface without changing the underlying data. Thus, a document can change to adapt the needs and preferences of individual users, the author's original structure being retained. Such a dynamic adaptation technique has been implemented in the Petri-net based Trellis system developed by Stotts and Furuta.

6. Linearization of Hypertext

The reverse problem of converting text to hypertext is to linearize a hypertext document for printing. The need for a linear, printed document will exist for some time to come. Yankelovich et al. suggest that "printing a branching [hypertext] document in a linear fashion poses both technical and conceptual problems." [Yankelovich et al., 1985]. It is easy to linearize a hypertext document having a strict hierarchical structure by performing a depth-first tree traversal, by printing the first chapter and its sections and moving onto the next chapter and so on. However, in the general case where the hypertext document is a highly connected network without any special order, it is very difficult to produce a good linear document [Nielsen, 1990].

Activities involved in the production of a linear document are together referred as "document preparation" [Trigg & Irish, 1987]. According to Trigg and Irish, document preparation does not include writing activities such as notetaking and reorganizing. It can happen throughout the writing process, often commencing well before the final text is composed. Paper structuring or layout is done in an outline, which is massaged and fine-tuned for some time before any text is written. In NoteCards, the outline takes the form of a filebox hierarchy whose filebox titles correspond to section titles of the paper. Some writers just use a single text card to capture the overall structure of the paper to be written. This may involve pulling up other relevant cards and copying text from these note or paraphrasing. This kind of a composition of text (essentially in a linear fashion) is the same as moving away from the concept of hypertext. Smoothing the document and integration is done with the linear document rather than with the source cards.

The connection between the linear form and the hypertext document can be maintained over time. This can be done using document cards which allow users to automatically generate a linear document, in a card, covering some portion of the network. Changes to the document are made in the source cards from which the document was compiled. This allows portions of the paper to be visible in different windows and simultaneously accessible [Trigg & Irish, 1987].

The SmarText Electronic Document Construction Set, a software product that automates the creation and browsing of large hypertext document, presents multiple views of non-linear text in a linear fashion. SmarText readers can choose to traverse one path out of many possible paths. A path is essentially a linear presentation of specific nodes connected by specific links. The text, the index and outlines are constrained by the selected view or path [Rearick, 1991]. The concept of paths has also been explored using the Scripted Document System at Xerox [Zellweger, 1989]. A path can also be used to collect all interesting documents to form a linear document that can be printed [Utting & Yankelovich, 1989]. Another possible method of linearizing hypertext is to take the user’s history of interaction and print the contents of the nodes that were traversed during a particular session.

7. Summary

Converting linear text to hypertext has been a classic problem while dealing with large information spaces such as encyclopedias, training manuals and dictionaries. Attempts have been made to convert these printed material both by manual and automatic means. Some researchers have suggested guidelines for conversion. In addition to automatic conversion of text to hypertext based on structural features, researchers have attempted to construct automatic links based on lexical and semantic analysis of text. Hypertext templates facilitate the design, organization, and presentation of a collection of knowledge in the form of hypertext. Researchers have suggested some general guidelines for authoring hypertext documents. These include splitting a hypertext document into three components: the content part, the organizational part, and the presentation part. There have been efforts to make hypertext systems more dynamic by incorporating virtual structures, computation, and filters. Though miniscule in nature, attempts have been made to linearize hypertext documents for the purpose of printing.

References

[Bieber, 1991]. Bieber, Michael. Issues in Modeling a "Dynamic" Hypertext Interface for Non-Hypertext Systems, Proceedings of Hypertext’91, ACM Press,1991.

[Bieber, 1993]. Bieber, Michael. Providing Information Systems with Full Hypermedia Functionality, Proceedings of the Twenty-Sixth Hawaii International Conference on System Sciences, 1993.

[Bernstein, 1990]. Bernstein, Mark. An Apprentice That Discovers Hypertext Links, Proceedings of

ECHT ’90, 1990.

[Catlin et al., 1991]. Catlin, Karen S., Garrett, N.L., and Launhardt, Julie A. Hypermedia Templates: An Author’s Tool, Proceedings of Hypertext ’91, ACM

Press, 1991.

[Cook, 1988]. Cook, Peter. An Encyclopedia Publisher’s Perspective, Interactive Multimedia, Apple Computer Inc., Microsoft Press, 1988.

[Glushko, 1989]. Glushko, Robert J. Transforming Text Into Hypertext For a Compact Disc Encyclopedia, Proceedings of CHI ’89, ACM Press, 1989.

[Halasz, 1988]. Halasz, Frank. Reflections on NoteCards : Seven Issues for the Next Generation of Hypermedia Systems, Communications of the ACM, July 1988.

[Nielsen, 1990]. Nielsen, Jakob. Converting Existing Text to Hypertext, Chapter 11, Hypertext and Hypermedia, Academic Press, 1990.

[Rao & Turoff]. Rao, Usha & Turoff, Murray. Hypertext Functionality: A Theoretical Framework, International Journal of Human-Computer Interaction, 1990.

[Raymond & Tompa, 1988]. Raymond, Darrell R., and Tompa, Frank WM. Hypertext and the Oxford English Dictionary, Communications of the ACM, July 1988.

[Rearick, 1991]. Rearick, Thomas C. Automating the Conversion of Text Into Hypertext, Hypertext/Hypermedia Handbook, Eds. Berk, E. and Devlin, J., Intertext Publications/McGraw Hill Publishing Co., Inc., New York, 1991.

[Riner, 1991]. Riner, Rob. Automated Conversion, Hypertext/Hypermedia Handbook, Eds. Berk, E. and Devlin, J., Intertext Publications/McGraw Hill Publishing Co., Inc., New York, 1991.

[Stotts & Furuta, 1991]. Stotts, P.David, and Furuta, Richard. Dynamic Adaptation of Hypertext Structure, Proceedings of Hypertext ’91, ACM Press, 1991.

[Thuring et al., 1991]. Thuring, Manfred, Haake, Jorg M., and Hannemann, Jorg. Hypertext ’91 Proceedings, 1991.

[Trigg & Irish, 1987]. Trigg, Randall H., and Irish, Peggy M. Hypertext Habitats: Experiences of Writers in NoteCards, Proceedings of Hypertext ’87, ACM Press, 1987.

[Yankelovich et al., 1985]. Yankelovich, Nicole, Meyrowitz, Norman, and van Dam, Andries. Reading and Writing an Electronic Book, IEEE Computer, 1985.

[Zellweger, 1989]. Zellweger, Polle T. Scripted Documents : A Hypermedia Path Mechanism, Proceedings of Hypertext ’89 Conference, November 1989.

CHAPTER 3

DATABASE ISSUES

1. Introduction

Most current hypertext systems use proprietary data formats for storage. Some researchers have tried using Relational Data Base Management Systems (RDBMS) to implement hypertext applications [Schutt & Streitz, 1990], [Meyrowitz, 1986]. There exists a large class of applications for which relational database management systems and other more conventionally structured database systems are too limited. These applications can be characterized as complex, large-scale, data-intensive programs such as CAD/CAM systems, documentation management systems, hypermedia systems, and geographical information systems. This class of applications needs a database model that is more expressive and flexible than the relational model. Some researchers believe that object-oriented databases can meet the data handling requirements of such applications.

Among the many research issues in the area of hypermedia, database requirements of hypermedia systems have received very little attention. HyperBase, based on an object-oriented data model, was implemented on Sybase, a RDBMS. Similarly, Intermedia was originally implemented on INGRES and later on C-Tree. Very little work has been done in the usage of Object-Oriented Data Base Management Systems (OODBMS) for hypermedia applications. The use of an OODBMS for a hypermedia system is an excellent marriage between the rigorous infrastructure and modeling abilities of object-orientation and the flexibility and navigational access to information characterized by hypermedia [Lange, 1993].

2. Object-Oriented Concepts and Hypermedia

Before we discuss the database requirements of hypermedia, let us try to understand the object-oriented paradigm and analyze its relevance to hypermedia. The advantages of using an OODBMS includes the ability to model complexity, to provide a unified representation of the problem domain, and to build complex objects.

There have been object-oriented approaches to systems analysis and design, programming languages, operating systems and database systems. Object-oriented systems provide many new features that are not present in conventional structured systems. The object-oriented paradigm more closely represents the problem domain by mapping abstractions of real world entities as objects/classes. It also focuses on the definition and inheritance of behavioral capabilities in the form of operations embedded within objects. The paradigm also supports simpler capabilities for structuring complex objects.

Object-oriented systems, including languages and database management systems support concepts such as objects, classes, methods, data abstraction, encapsulation, inheritance, schema evolution, reusability,concurrency control, correctness, polymorphism, composite objects, active objects, version control and direct manipulation [Cattell, 1991].

Many features from the world of object-oriented analysis, design and programming can be extended to the hypertext model.

1. Simple nodes can be compared to atomic objects representing primitive data types such as integer, character, string, video frame and bitmap.

2. Objects or nodes can be accessed using object identifiers or node identifiers.

3. A link can be represented by a set of at least two object identifiers. Links can also be treated as objects with their own identifiers (link identifiers) which can be used to separate index information from content.

4. A composite node in hypertext (made of webs of nodes and links) can be treated as a composite object or an aggregation of simple objects.

5. The concepts of data abstraction and encapsulation can be applied by defining methods to create, delete, update and manipulate nodes and links, to traverse links, and to trigger events.

6. Nodes and links can be grouped under different classes based on structural and behavioral patterns (semantics). Organizing nodes and links semantically helps manage the network better, eliminates ambiguity, clearly differentiating the purposes of these objects.

7. Nodes and links of a particular class can also inherit properties from related superclasses. This feature can be used in the creation and management of hypermedia templates - when a user changes a parent template, these changes can be propagated to all its sub-classed templates.

8. Whenever the properties of nodes and links have to be changed it should be easy to do so through schema evolution.

9. Other object-oriented concepts that are pertinent to hypertext include concurrency control, versioning, and persistence.

In HyperBase, the basic class was a HB_Object which had three subclasses: HB_Node, HB_Link, and HB_Composite_Object [Schutt & Streitz, 1990]. HB_Nodes are HB_Objects with content and history. HB_Links are HB_Objects that connect two existing HB_Objects. HB_Composite_Objects are collections of references to existing HB_Objects. The generic operations of create, modify, copy, delete, retrieve an object (part of or whole) were supported for nodes, links, composite objects, and attributes. Since the data model was object-oriented, a natural choice to implement HyperBase would have been an OODBMS. However, for reasons of availability, a first prototype was implemented on Sybase, a RDBMS.

3. Requirements for Next-Generation Hypermedia Systems

Frank Halasz had originally identified seven issues for next generation hypermedia systems [Halasz, 1988]. Other researchers have come up with a few more over the past few years [Lange, 1993], [Fountain et al., 1990]. Object-oriented database management systems seem to meet most of these requirements.

a. Openness and Distribution

Most current hypertext systems are basically closed systems. That is, material created in one system cannot be easily integrated with material created in another system. An open hypermedia system is a system which can connect to other information systems (both hypertext and non-hypertext). It should be able to freely exchange data. Distribution allows the system to store at geographically dispersed sites in a manner transparent to the user.

b. Support for collaborative work or sharing

Future generation hypermedia systems must provide adequate support for collaborative work. This includes simultaneous multi-user access to the hypermedia network, robust concurrency control mechanisms, broadcasting to users any changes made to the network by other users, and tracking contributions made by each member of a team.

c. Data integrity/Correctness

The database layer should preserve data integrity and provide traditional secondary storage management and data administration facilities. It should forbid dangling references to objects or attributes and prevent recursive inclusion of composite objects [Schutt & Streitz, 1990]. It should also support either the notion of rules as in extended relational databases or the concept of semantics as in object-oriented databases.

d. Dynamism

Most current generation hypermedia systems deal with static information. Future systems need the ability to dynamically reconfigure the network in response to changes made to the network or its contents. Virtual or dynamic structures (active objects) are similar to dynamically generated views in relational databases. Virtual structures are possible only when current models are changed to adapt to changing information.

e. Search and query mechanism

In addition to navigational access to information, hypermedia systems should provide efficient search and query mechanisms. This would partly solve the "lost in space" (disorientation) problem experienced by users during navigation. There can be two kinds of queries - a structure query to retrieve a part of the network and a content query to retrieve a specific node. The former requires the development of a query language geared towards dealing with hypertext network structures.

f. Computation

The integration of AI into hypermedia engines will be an interesting area to explore. This would include inheritance, truth maintenance, rule based searching, and inference engines. Computation built into the hypermedia system is likely to be more efficient for extensive access to information in the network.

g. Composites

With increasing use of hypermedia for sophisticated applications it becomes necessary to deal with groups of nodes and links. This will require making a composite node as a primitive construct in the basic hypermedia model. Inclusion (or part-of) relations have to be supported in addition to standard, referential links.

h. Versioning

Versioning is required in order to keep track of changes to the network. This can include versioning at the level of individual entities such as nodes and links and also at the level of the hypermedia network as a whole.

i. Multimedia Support

The database layer should be able to efficiently store and retrieve multiple media. It should also provide transparent access to different storage media.

j. Extensibility and Tailorability

Extensibility involves the ability to handle extensions to the existing data model (schema evolution) in a flexible and safe manner. The database layer should not only be able to handle the structural part of the hypertext data model but also the semantic part to ensure data abstraction and encapsulation of the evolving data model.

4. Object-Oriented Databases and Hypermedia

Intermedia, developed at Brown University, is an object-oriented hypermedia system implemented on a relational database management system, INGRES. In an experimental study conducted by researchers at the University, INGRES was compared with an object-oriented database management system (Encore, developed in-house) for the implementation of Intermedia [Smith & Zdonik, 1987].

RDBMS	OODBMS
Hierarchies to be flattened into relations	Hierarchies can be represented as they are.
Objects retrieved through query language	Operations on objects stored in the database
Many queries to retrieve a single object	One message sent to an object which can send messages to other objects
Results from queries to be stored in application’s data structure	Messages can directly manipulate data in the database and return a pointer to the result
High level of data structure and impedance mismatch	Lower level of impedance and data structure mismatch
Integrity checks enforced only during commit time	Integrity checks performed during run-time through triggers
Due to the flat nature of relations	An entire hierarchy can be locked in

each record in each relation in a given hierarchy to be explicitly locked	a single operation

Table 3.1 - Results of Evaluation at Brown University [Smith & Zdonik, 1987].

At the conclusion of the study, researchers felt that an object-oriented database could successfully meet the data handling requirements of Intermedia.

In a more recent study conducted by researchers at the University of Tokyo, four different types of storage mechanisms were evaluated against some of the requirements listed in Section 2. The following table below shows the results of the evaluation.

	Filesystem	KnowledgeBase	RDBMS	OODBMS
Openness	Partly	No	Yes	Yes
Sharing	Partly	No	Partly	Yes
Integrity	No	Yes	Yes	Yes
Multimedia	No	No	No	Yes
Querying	Partly	Yes	Yes	Partly
Versioning	Partly	No	No	Partly
Extensibility	No	Yes	Yes	Yes

Table 3.2 - Results of Evaluation at University of Tokyo [Lange, 1993].

From Table 3.2, it is apparent that an OODBMS scores high on all aspects of the reqirements for a storage mechanism for hypermedia systems. Without going into details on the file system approach and the knwledge base approach, let us compare the RDBMS approach against the OODBMS approach (Table 3.3 below).

RDBMS	OODBMS
Easy to implement since concepts are well known	Technology is new and hence difficult to implement
Bad performance even for simple link traversals	Navigation model of the database can be directly exploited
Object-identity and composite objects to be directly modeled in the hypertext model rather than the database	Commonalities between the database and the hypertext model facilitate direct implementation of data structures and behavior
Complex data to be flattened leading to loss of data abstraction, encapsulation and performance	Efficient data handling mechanisms for complex data
Not very good support for collaborative work such as version management and concurrency control	Flexible transaction processing facilities and wide range of locking policies

Table 3.3. - Specific Differences between an RDBMS and an OODBMS [Lange, 1993].

As an outcome of their experimental studies, this research team is investigating into an object-oriented extension to the hypertext data model. The cross-breeding of object-orientation and hypertext is called object-oriented hypermodeling. This concept has been utilized in a tool for rapid object-oriented application development on an object-oriented database. The tool supports the hypermodeling process through a graphical user interface, a C++ class library and Object SQL on the object-oriented database ONTOS DB. The combination of object-oriented hypermodeling and ONTOS DB together satisfied all the requirements listed in Section 2 except version control [Lange, 1993]. Some recently released OODBMS such as Versant, Statice, and O2 have been used in developing hypermedia applications [English, 1992].

In summary, these studies have shown that object-oriented database management systems are more appropriate as storage mechanisms for hypermedia systems than traditional methods. Researchers are also working on a new class of systems called hyperbases. A hyperbase is a storage mechanism with a built-in hypertext data model providing primitives to manipulate instances of this model. It has also been proposed that by integrating an object-oriented data model with hypertext, the information retrieval process can be greatly improved [Rao et al., 1993]. This can be accomplished by redirecting hypertext functionality from the application level to the database level using object-oriented database systems.

5. Summary

Object-oriented technology provides the ability to model, to develop and to manage complex systems that cannot be easily implemented with current technology. Hypermedia systems constitute an emerging class of complex management information systems. They should facilitate distributed memory, distributed processing, and distributed knowledge on a network. An object oriented approach to hypermedia may help us achieve this goal much better than current approaches. However, research in this area is still in its infancy and there are many open issues to be investigated. Any breakthroughs in this arena will take us closer to realizing Ted Nelson’s goal, "Everything should be available to everyone. Any user should be able to follow origins and links of material across boundaries of documents, servers, networks, and individual implementations. There should be a unified environment available to everyone providing access to this whole docuverse." [Nelson, 1991].

References

[Cattell, 1991]. Cattell, R.G.G. What are Next-Generation Database Systems ? , CACM, October 1991.

[English, 1992]. English, Larry. Object Databases at Work, DBMS, October 1992.

[Fountain et al., 1990]. Fountain, Andrew M., Hall, Wendy, Heath, Ian and Davis, Hugh C. MICROCOSM: An Open Model For Hypermedia With Dynamic Linking, Proceedings of ECHT ’90, 1990.

[Halasz, 1988]. Halasz, Frank. Reflections on NoteCards: Seven Issues for Next Generation Hypermedia Systems, CACM, July 1988.

[Lange, 1993]. Lange, Danny. Object-Oriented Hypermodeling of Hypertext Supported Information Systems, Proceedings of HICSS ’93.

[Meyrowitz, 1986]. Meyrowitz, Norman K., Intermedia: The Architecture and Construction of an Object-Oriented Hypermedia System and Applications Framework, OOPSLA ’86 Proceedings.

[Nelson, 1965]. Nelson, Ted. A File Structure for The Complex, The Changing and The Indeterminate, ACM 20th National Conference, 1965.

[Nelson, 1991]. Nelson, Ted. Speech delivered at MultiMedia Expo ’91, New York.

[Rao et al., 1993]. Rao, Gururajan R., Balasubramanian, V., and Suresh, B.S. Integration of Hypertext and Object-Oriented Databases for Information Retrieval, Proceedings of the Nineteenth Annual Northeast IEEE Conference on Bio-Engineering, IEEE Press, March 1993.

[Schutt & Streitz, 1990]. Schutt, Helge A., and Streitz, Norbert A. HyperBase: A Hypermedia Engine Based on a Relational Data Base Management System, Proceedings of ECHT ’90, 1990.

[Smith & Zdonik, 1987]. Smith, Karen and Zdonik, Stanley. Intermedia: A Case Study of the Differences Between Relational and Object-Oriented Database Systems, OOPSLA ’87 Proceedings.

CHAPTER 4

USER INTERFACE ISSUES

1. Introduction

The promise of hypertext lies in its ability to produce complex, richly connected and cross-referenced bodies of information. However, it can also become a complex system of tangled webs, confusing both authors and readers. According to Conklin, disorientation and cognitive overhead are the two most challenging problems related to hypertext. He feels that these two problems "may ultimately limit the usefulness of hypertext." [Conklin, 1987].

1.1 Disorientation

The problem of disorientation or "getting lost in space" arises from the need to know where one is in the network, where one came from, and how to get to another place in the network. In traditional text, it is not easy to get lost. There is the table of contents of topics with page numbers, the index with keywords and page numbers, and also bookmarks. However, in a complex hypertext network, with thousands of nodes and links, it is more than likely that the reader will get lost.

1.2 Cognitive Overhead

Cognitive overhead is the additional mental overhead on authors to create name, and keep track of nodes and links. For readers, it is the overhead due to making decisions as to which links to follow and which to abandon, given a large number of choices. The process of pausing (either to jot down required information or to decide which way to go) can be very distracting. It can become a serious problem if there are a large number of nodes and links.

All hypertext systems provide the basic capability of following a uni-directional link to a target node. However, the true potential of hypertext cannot be realized by this approach alone. Considerable amount of research efforts are underway in universities and the computer industry to develop better tools and methods to exploit the full potential of hypertext and also to solve or minimize the problems of disorientation and cognitive overhead.

2. Designs for Navigation

2.1 Graphical Browsers

Graphical browsers serve as overview displays for large bodies of information, especially in a hypertext system. NoteCards from XEROX and gIBIS from MCC both provide these browsers with which users can scroll through the entire network as well as rearrange the nodes. Both systems provide facilities to view the contents of the browser at different levels of detail. For large networks, the highest level of detail about the structure will be provided. The user can zoom in to see any portion of the browser in detail but cannot look at the details of the entire network because of screen space limitations [Utting & Yankelovich, 1989].

Graphical browsers help reduce disorientation by providing a two-dimensional spatial display of the hypertext network. They also help minimize cognitive overhead by showing a small part of the network. They also provide an idea about the size of the network which help users estimate the number of nodes and links in the system. If the user needs an idea about the contents of a target node prior to visiting it, some kind of a "peek" mechanism can be provided. Though general browsing is a low cognitive load operation, it is inefficient for directed search tasks or fact retrieval. Also, a graphical browser itself can become a tangled web if the hypertext network is large and if the display has to be updated because of dynamic changes in the network.

2.2 Web Views

In Intermedia, a hypertext system developed at Brown University, a web is defined as a network of documents or portions of documents linked together. Initially, the designers provided users with Web Views (similar to graphical browsers) called global map, local map, and local tracking map. A global map displayed every document in a web and the links between them. However, a global map worked well only with small webs. The local map was useful in focusing on the document of interest and its neighboring documents. The local tracking map dynamically updated the focus as documents were opened and activated.

Studies showed that the global map and the local tracking map were not of much use to users. Disoriented users need a sense of context and location. The need was felt to display no only spatial information ("Where can I go from here ?") but also temporal information ("How did I get here ?") [Utting & Yankelovich,1989]. The Web Views were modified to provide both spatial and temporal context in a flexible and non-intrusive manner. Three navigation tools were developed: a path, a map, and a scope line. Most often users would like to know, in advance, the amount of material in the web. This would help them decide whether to continue reading a document or to return later. The scope line informs the user about the number of documents and links in the web. Paths and maps are discussed in other sections below.

2.3 Maps and Overview Diagrams

Maps serve to improve spatial context in a hypertext network. The Intermedia view of a map is a local tracking map that displays all the documents or nodes linked to the current document which is dynamically updated. This ensures that correct information is always available; it also allows link previewing. Selecting a link marker will highlight the corresponding link line allowing the user to get an idea of the target without actually following the link [Utting & Yankelovich, 1989].

Overview diagrams, both at the local and global levels, serve as excellent navigational aids. Global overview diagrams provide an overall picture and can also serve as anchors for local overview diagrams. Local overview diagrams provide a fine-grained picture of the local neighborhood of a node. Overview diagrams for large systems might become complex and might introduce navigational problems of their own [Nielsen, 1990a]. Nielsen feels that in order to reduce the propagation of links from the local diagrams to global diagrams, weights can be assigned to the links based on their relevance to the user; this will trim the edges of the graph at the global level. Also, anchors can be differentiated, graphically, based on their estimated relevance to the users. Statistical analysis of the most frequently traversed links by a number of users might provide valuable insights into developing system-generated overview diagrams.

2.4 Paths and Trails

In most current hypertext systems, readers have a problem trying to understand the material presented because they view it in the wrong order or they simply cannot comprehend easily. The concept of a path can help solve this problem by allowing authors to determine an appropriate order of presentation for a given audience. It will reduce both disorientation and cognitive overhead since users will follow a pre-defined path which will also narrow down their choices. The concept of paths or trails was first enunciated by Bush in his classic paper, "As We May Think". He called a trail a sequence of links through a memex.

"The process of tying two items together is the important thing.....Before him are the two items to be joined, projected onto adjacent viewing positions....The user taps a single key, and the items are permanently joined.....Thereafter, at any time, when one of these items is in view, the other can be instantly recalled merely by tapping a button....Moreover, when numerous items have thus been joined together to form a trail, they can be reviewed in turn, rapidly or slowly.....It is exactly as though the physical items had been gathered together from widely separated sources and bound together to form a new book. It is more than this, for any item can be joined into numerous trails....trails do not fade....Several years later,.....tapping a few keys projects the head of the trail. A lever runs through it at will, stopping at intersecting items, going off on side excursions." [Bush, 1945].

The idea of trails was implemented as paths by Trigg in his TEXTNET system. In Intermedia, a path is a list of documents users visited earlier in a browsing session. The display of a path consists of the name of the document, an icon indicating the type of event (opening or activating documents), and a timestamp indicating when the event occurred. A user’s path is saved when closing the web and restored when opening the web, the next time. Thus, a path can be used to collect all interesting documents to form a linear document that can be preserved in printed form [Utting & Yankelovich,1989].

The Scripted Documents System, developed at XEROX, uses paths which are procedural and programmable. The items in the path (nodes) can be "active" entries or scripts which can do computations, execute programs etc. The entries provide the content while the path provides sequencing. Paths can be created and edited using path editors. Readers can get local and global views of relevant paths. Playback mechanisms are supported which allow users to play back a path either single-stepped or automatic. Different kinds of scripts provide different paths and can be used to create presentations for different classes of audiences [Zellweger,1989].

2.5 Guided Tours and Tabletops

The concept of paths was extended to "guided tours" in NoteCards [Trigg,1988]. A guided tour is a system-controlled navigational tool that can be entered and exited at the user’s will. Progress can be monitored using maps or overviews of the hypertext database. In NoteCards, a guided tour can be accessed through a graphical browser (which displays a portion of the hypertext network in terms of cards and links) by both authors and readers; the "stops" on the guided tour are sets of cards arranged on the screen according to a particular layout, also called the "tabletop". Thus, a guided tour is a graphical interface to a network or path of tabletop cards, connected by links. A tabletop is a snapshot of cards currently on display, including their positions, shapes, scrolled locations of their contents and any overlapping. The decision about what cards to include in the tabletop is left to the author. Tabletops, in addition to saving individual card layouts, can be used to make online presentations and hard-copy screen shots or transparencies for offline presentations. Editing tools are available for creating and managing both tabletops and guided tours.

A reader has necessary controls to start a guided tour, traverse a path of tabletops in sequence, jump arbitrarily to any tabletop, go back and forth between tabletops or reset the state of the guided tour. Facilities are also available to peek into tabletops in a path without actually "opening" them. Guided tours, I would add, are like "hypertext within hypertext". They help bridge the communication gap between hypertext authors and readers. They make the contents of a document intelligible to people who are not already familiar with it - things like a description of their contents, the rationale behind their contents, an explanation about the context of their use and a history of their construction. A demonstration can be carried out by the author, in absentia, using guided tours. Trigg refers to this advantage of remote pointing as "remote deictic reference".

Nielsen feels that even though a guided tour seems to limit the very purpose and potential of hypertext, it can be used to introduce the concept of hypertext to new readers and for small systems. Different levels of guided tours can be designed for different types of users [Nielsen, 1990b].

2.6 Backtracking, History Lists, Timestamps, and Footprints

Backtracking allows visiting previously visited nodes. Backtracking in a linear fashion and also the ability to arbitrarily jump to previous nodes helps people to bail out of difficult situations. The most preferred method is the path-following principle which allows traversing, in reverse order, those nodes that were previously visited since this approach relies on the user’s memory of his or her own navigation behavior. The structure-oriented feedback approach allows users to directly jump to a node without backtracking. However, experiments have shown that combining these two methods might lead to confusion [Nielsen, 1990a].

Backtracking has been provided in system such as Guide and HyperCard. However, most backtracking mechanisms do not support the feature of going forward to the node from which the backtracking was initiated. Also, backtracking can confuse users if the interface is not consistent. In a document designed with Guide when readers returned from a backtracking operation the display looked different making them wonder whether that was the node they were viewing before taking off on the backtrack [Nielsen, 1990b]. Hence, Nielsen suggests that backtracking mechanisms must fulfill two requirements - "It should always be available, and it should always be activated in the same way."

A textual history list mechanism was developed for NoteCards which maintains an ordered list of each notecard that was examined in a particular session. Users can select an item from the list and look through a browser. Nodes that have already been visited in a session are marked with a plus sign, one for each time visited. Visual indicators such as the plus sign or checkmarks or asterisks serve as "footprints" on overview diagrams and help users to avoid returning to nodes that have been recently visited. Some implementations of the history list allow users to customize the list for future interaction. A variation of the history list called the history tree shows the users "how" they traversed a set of linked nodes, the digressions, and multiple visits to nodes. Both the history list and history tree can be saved and annotated with text and graphics [Utting & Yankelovich,1989].

HyperCard has a graphical history list called the recent list which has miniature snap-shots of the last forty-two nodes visited. Clicking on a miniature brings that card to the display. Of course, this method makes the assumption that an individually may not be able to remember the name of the node but may remember the "look" of the node. Some usability studies have shown that it is very difficult for users to distinguish these miniatures from one another [Nielsen, 1990a].

Nodes that are visited can be timestamped (along with the accumulated time spent at each node) and maintained in a chronological order [Nielsen, 1990a]. This feature will help users, at a later date, to recognize whether the node was visited before and if so, the duration allowing users to change their viewing behavior accordingly. The Document Examiner supports a history of commands executed and a history of documents examined. The command history allows people to see previously executed commands and the document history is in the form of bookmarks for each document that was previously opened.

2.7 Arbitrary Jumps, Landmarks, and Bookmarks

Arbitrary jumps or gotos can be provided enabling users to go to any node in the system. This can be accomplished by zooming in and out using an animated iris that opens for anchored jumps and closes for return jumps. "Landmarks" or prominent nodes can be provided which can always be accessed from anywhere in the system. The concept of a bookmark is similar to a history list except for the fact that a bookmark is placed by the reader only if a particular node will be of interest at a later date. Hence, a bookmark list is smaller more manageable and more relevant to the user.

Experiments have shown that random jumps from anchors leading to multiple destinations can be very confusing to users [Nielsen,1990a]. This can be avoided by listing the possible destinations when an anchor is activated allowing the user to choose a predictable path. In Intermedia, this problem has been solved by displaying a single link icon (instead of multiple links) which can be quickly queried to show the specific links, their names, and their destination nodes.

2.8 Embedded Menus

Embedded menus, as opposed to explicit menus, allow the user to select a word or item embedded within the text of a node and can be selected using a touch screen, cursor keys or mouse pointer. In The Interactive Electronic Encyclopedia System (TIES), selectable items are highlighted directly in the text, a method now called touchtext [Koved & Shneiderman, 1986]. The embedded menus in TIES (now called HyperTIES) can be traversed using cursor keys and can be selected either by clicking or by a keystroke; a new node or piece of information will be retrieved. At every node in the network, the user can request a return to a previous article using other navigation mechanisms. An extension of the embedded menus approach can be found in MIT’s Spatial Data Management System (SDMS) where users can select a graphical area (such as a territory in a map) and retrieve a more detailed map from the database. The user can undo the effects of the selection process by returning to less detailed maps.

Embedded menus are a better way of indexing for hypertext systems since they emphasize the understanding of concepts. They highlight semantic relationships rather than physical relationships. They provide meaningful task domain terms (as opposed to computer domain terms) and concepts, thereby reducing disorientation [Marchionini & Shneiderman, 1988]. Embedded menus reduce the "loss of context" feeling by being part of the information being displayed. They provide information hiding (layers below are shown only when requested). The extent of "embeddedness" can be varied depending on the skill level of the user. They are suitable for learning-by-browsing systems as in museums. Further research is required regarding the negative aspects of using highlighted embedded menus. It is possible that they may cause disruption, reducing speed and comprehension.

2.9 Fisheye Views and Spiders

Furnas implemented a fisheye view algorithm which is similar to looking at a scene with a wide angle lens - things of greater interest will be at the center, while items of lesser interest will be on the periphery. This algorithm generates an image of the neighborhood by computing a relationship between a priori importance of a node and the distance between that node and the current position in the hypertext network [Carlson, 1989].

Thoth-II treats hypertext as a directed graph with semantics [Collier, 1987]. Nodes themselves do not contain text. Pieces of text are connected to the nodes by text links while the nodes are connected to each other by labeled value links; text can contain embedded links, called lexical links, to other nodes. "Spiders" is a directed graph browser in Thoth-II where a global map is created dynamically as a user browses through linked nodes. As the user interacts with the structure that is being viewed, new graphic objects (nodes and links) are created. Activating a node expands it showing links to other nodes which themselves fan out to other nodes. Thus, the network expands or fans out in two-dimensional space creating "spiders" on the display. The window on which this structure is displayed can be moved around to view other parts of it falling outside the viewing area. The browsing mode allows the user to browse through the graph and manipulate nodes and links whereas the text mode allows the user to view the textual pieces attached to the nodes. The disadvantage with this system is that the whole screen can get filled up with spiders very quickly as the user clicks on various nodes.

2.10 Roam and Zoom Techniques

Hypertext navigation is also restricted by the physical limitations of the display screen. The inability to view large amounts of information at any one time due to the small size of computer displays falls under the category of context-in-the-small problems [Nielsen,1990a]. This includes the issue of readability, when the contents of a node do not fit into one screen and have to be carried over to other screens. Larger displays can only partially solve this problem. Conventional scrolling techniques use arrow keys or paging keys allowing only vertical movement to various places on the screen. While scroll bars do allow two-dimensional navigation, it is not easy to focus on a particular region of interest.

In an effort to improve the display of large two-dimensional spaces, researchers at the University of North Carolina, Chapel Hill, developed two similar direct manipulation techniques. Both use a miniature of the entire information space to assist the user in remembering his or her location. The first technique allows the user to rapidly "roam" over the space while the second allows "zooming" into a particular region while roaming [Beard & Walker, 1987].

The entire information space can be shown, in miniature form, in a map window occupying a small part of the display. A wire-frame box or rectangle inside the window shows the portion of the information space displayed on the main display. The main display is the actual viewport into the information space.

Thus, the map window provides a clear sense of location within the information space. the size of the wire-frame can be changed using the mouse thereby zooming in and out of the region. Also, the wire-frame box can be dragged around the map window thus roaming around the information space. Experiments proved that the roam and zoom features were significantly faster than vertical and horizontal scroll bars thus improving performance.

2.11 Conceptual Space Navigation

Some researchers feel that learning systems should have some amount of disorientation and cognitive overhead in order to facilitate exploration and learning [Mayes et al., 1990]. They feel that while most researchers are concentrating on navigation through information space, very little work has been done on navigation through "conceptual space". They contend that simply following links to nodes does not necessarily provide effective learning - "they do not tell us ABOUT anything, but only WHERE it is."

In learning systems, disorientation in conceptual space is required sometimes in order to explore and learn. Thus, users need to be guided not only by system information but also by discovery. The issue of cognitive overhead has not been found significant in learning systems [Mayes et al., 1990]. The authors say, the very question "what to do next, will enrich the process of learning rather than detract from it." A learning-by-browsing system called StrathTutor was developed where the links were computed based on how the nodes were related to each other, conceptually, in terms of certain attributes. This not only allows navigation through conceptual space but also the ability to "interrogate" the system by designating a combination of attributes that are meaningful at a particular instant of exploration; the system will respond by giving the learner a guided tour of all nodes satisfying those attributes.

StrathTutor also provides another facility called the "quiz" where the learner is asked to play a game in which he or she is asked to identify the nodes that have common attributes. This helps the learner create his or her own personal view of the underlying conceptual space. The content search and structure search mechanisms, suggested by Halasz, will be especially useful in learning systems.

3. Usability and Evaluation of Hypertext

Wright feels that the following five issues have to be examined while evaluating hypertexts [Wright, 1991]:

1. Adequacy of the content and the interface.

2. Acceptability to readers.

3. Adaptability by readers for the task in hand.

4. Skills of the readers as information users.

5. Costs of production and dissemination.

3.1. Adequacy of the content and the interface

Hypertext design involves a mixture of decisions relating to the content, functionality, display, and control. The usefulness of hypertext depends on its purpose, ease of navigation, and the population domain. The authors have to determine tradeoffs among various design principles. For example, for a learning system the authors would have to not only look into the outcomes of using hypertext but also the quality of learning. In short, there is no upper limit on adequacy.

3.2. Acceptability to readers

There is also the issue of adequacy once the users have learnt to accept a hypertext system. For people seeking more information (say, after an initial exposure to a tourist information kiosk), the restrictive design of the application may pose a limitation (say, the tourist would like to know, immediately, about some good restaurants near a place of historic interest) - in fact, the display may not be large enough to display more details without chunking information. An overview mechanism (for browsers) and a query mechanism (for information seekers) may be more appropriate in such a situation.

3.3. Adaptability by readers for the task in hand

Hypertext systems have to be appropriate to the tasks that the users are trying to accomplish. There should be facilities for annotation and placing bookmarks. Hypertext interchange (the compatibility of file formats) is an important criterion of adaptability and hence acceptability. With the increasing use of hypertexts for collaboration, their evaluation should include the assessment of benefits to the group as a whole as well as the individuals.

3.4. Skills of the readers as information users

Hypertext evaluation should include an evaluation of the demand a system may make on the reading skills of the users. This could mean the evaluation of users reading linear text (on print). The system should extend the range of cognitive activities that readers will engage in (that is, provide an insight into strategies which they may not have thought of). The following could be some cognitive extensions :

Ability to search using keywords (general IR).

Changing the very nature of the work being done by creating some sort of an interdependency between the system and the users.

Evolution of new ways of thinking with the continued use of hypertext.

3.5. Costs of production and dissemination

The cost effectiveness of having information in hypertext form should be considered. Conventional IR systems allow only retrieval while hypertext systems help users integrate units of information retrieved.

Evaluation should also include looking for the availability of automated tools or cognitive prostheses to overcome cognitive overhead and disorientation. Readers adapt different navigation strategies based on the complexity of the task at hand. Such tools might help with a variety of subtasks related to hypertext usage. These subtasks might include specifying search targets and planning the order in which information will be sought. Tools are also required to store and manipulate the information found in order to integrate with other work. However, the effectiveness of such tools might be reduced due to the following :

Readers may not realize they need help.

Readers may not know how to use such tools.

Readers might blindly follow certain procedures without understanding their need.

Hence, the design and evaluation of hypertext systems should include the understanding of how people use their cognitive resources to handle information.

Nielsen also has discussed four categories for the evaluation of hypertext documents [Nielsen, 1991]. These are:

a. Utility

This is a measure of whether the hypertext document actually helps a user perform the intended task. This has to be compared with performing the same tasks with linear text. Many empirical tests have shown that readers exhibit poorer performance with hypertext than with paper documents.

b. Integrity

This is a measure of the completeness of the document - whether it is up to date and not misleading and if it is easy to maintain.

c. Usability

Usability can be measured as a combination of these factors:

Ease of learning - It is a measure of how fast a reader can start learning and navigating through a hypertext document. Good presentation structure and graphic design are key determinants.

Ease of use - It is a measure of how fast a reader can locate information. Appropriately defined links, search mechanisms, backtracks, landmarks and other navigational aids can greatly increase efficiency of use.

Error handling - It is a measure of how many errors a reader makes and how easy it is for them to recover from such errors.

d. Aesthetics

This is a measure of how pleasing is the system to the user.

4. Summary

In a true hypertext system, users must be able to move freely through the system according to their needs, without getting lost either spatially or cognitively. The facilities to navigate through a hypertext database must be at least as rich as those available in books. Some of the designs currently available to navigate through hypertext were reviewed in this chapter. Usability issues and evaluation criteria were discussed. When the initial excitement about hypertext dies down and systems become more common, better navigation techniques and more systematic evaluation measures will emerge both from developers and users. This will be based on the organizational setting, the targeted task domain, the typical user population, and the desired outcomes of navigation.

References

[Beard & Walker, 1987]. Beard, David & Walker, John. Navigational Techniques to Improve the Display of Large Two-Dimensional Spaces, University of North Carolina at Chapel Hill Technical Report TR87-031, November 1987.

[Bush, 1945]. Bush, Vannevar. As We May Think, The Atlantic Monthly, July 1945.

[Carlson, 1989]. Carlson, Patricia Ann. Hypertext and Intelligent Interfaces for Text Retrieval, The Society of Text, MIT Press, 1989.

[Collier, 1987]. Collier, George H. Thoth-II : Hypertext with Explicit Semantics, Proceedings of Hypertext ’87 Conference, November 1987.

[Conklin, 1987]. Conklin, Jeff. Hypertext : An Introduction and Survey, IEEE Computer, September 1987.

[Halasz, 1988]. Halasz, Frank. Reflections on NoteCards : Seven Issues for the Next Generation of Hypermedia Systems, Communications of the ACM, July 1988.

[Koved & Shneiderman, 1986]. Koved, Larry & Shneiderman, Ben. Embedded Menus : Selecting Items in Context, Communications of the ACM, April 1986.

[Marchionini & Shneiderman, 1988]. Marchionini, Gary & Shneiderman, Ben. Finding Facts vs. Browsing Knowledge in Hypertext Systems, IEEE Computer, January 1988.

[Mayes et al., 1990]. Mayes, Terry, Kibby, Mike & Anderson, Tony. Learning about Learning from Hypertext, Designing Hypermedia for Learning, NATO ASI Series, Volume F67, Springer Verlag, 1990.

[Nielsen, 1990a]. Nielsen, Jakob. The Art of Navigating through Hypertext, Communications of the ACM, March 1990.

[Nielsen, 1990b]. Nielsen, Jakob. Navigating through Large Information Spaces, Hypertext and Hypermedia, Academic Press, 1990.

[Nielsen, 1991]. Nielsen, Jakob. Panel Discussion - The Nielsen Ratings: Hypertext Reviews, Proceedings of Hypertext ’91, 1991.

[Trigg, 1988]. Trigg, Randall. Guided Tours and Tabletops : Tools for Communicating in a Hypertext Environment, ACM Transactions on Office Information Systems, October 1988.

[Utting & Yankelovich, 1989]. Utting, Kenneth & Yankelovich, Nicole. Context and Orientation in Hypermedia Networks, ACM Transactions on Information Systems, January 1989.

[Wright, 1991]. Wright, Patricia. Cognitive Overheads and Prosthesis: Some Issues in Evaluating Hypertexts, Proceedings of Hypertext ’91, 1991.

[Zellweger, 1989]. Zellweger, Polle T. Scripted Documents : A Hypermedia Path Mechanism, Proceedings of Hypertext ’89 Conference, November 1989.

CHAPTER 5

INFORMATION RETRIEVAL ISSUES

1. Introduction

Navigation or browsing is effective only for small hypertext systems. For large hypertext databases, information retrieval (IR) through queries becomes crucial. Conklin had suggested that search and query mechanisms can present information at a manageable level of complexity and detail [Conklin, 1987]. Halasz's view was that "navigational access itself is not sufficient. Effective access to information stored in a hypermedia network requires query-based access to complement navigation..........search and query needs to be elevated to a primary access mechanism on par with navigation." [Halasz, 1988].

2. Query and Search Mechanisms

Conventional IR systems focus on keyword based automatic searching (in conjunction with Boolean operations), weighting of words based on their statistical properties, ranking of documents according to probability of relevance, automatic relevance feedback for query modification and query languages [Croft et al., 1990]. However, very few (or none) of these methods retrieve complete or accurate information. Too general a query may yield a lot of items and too specific a query may retrieve no items. Thus, traditional IR is an inherently uncertain process. Combining inference techniques could eliminate or minimize uncertainty. In hypertext systems, a weighted keyword search combined with hypertext links can improve IR by finding only a subset of nodes or "hits" whose links can then be followed to other semantically related nodes [Carlson, 1989].

According to Halasz, query and search mechanisms can be classified into content search and structure search [Halasz, 1988]. Content search is standard IR technique extended to hypertext systems. That is, all nodes and links are treated independently and examined for a match to the given query. On the other hand, structure search will yield the hypertext sub-network that matches a given pattern. Query facilities which combine aspects of both content search and structure search will be capable of acting as filters. Based on the user's query, the interface will display only those nodes and links that match the query, filtering out other parts of the network. Filtered browsers have been implemented both for NoteCards and Tektronix's Neptune. In NoteCards, a user can filter out information based on the node or link type. In Neptune, the query can be content-based; if the query is broad enough, a global view of the entire network is displayed; if the query is well refined, the viewing size will be manageable.

2.1. Content Queries and Indexes

Bruza proposed a two-level architecture for hypertext documents, the top level called hyperindex (containing index information) and the bottom level hyperbase (containing content nodes and links) [Bruza, 1990]. The hyperindex consists of a set of indexes linked together. When an index term describing the required information is found, the objects from the underlying hyperbase are retrieved for examination. Navigating through the hyperindex (not the hyperbase) and retrieving information from the hyperbase is called "Query By Navigation" [Bruza, 1990].

An index is made of a set of index entries. Each index entry consists of a term descriptor or keyword and a locator (like a page number). Term descriptors lack specificity. Term phrases are made of term descriptors thus increasing specificity. However, they may retrieve too many items or no items at all and hence lack exhaustivity. Index expressions provide relationships between term descriptors. Thus, they are more specific than term phrase descriptors. Index expressions have a structure that can be used to derive a lattice of descriptors supporting query by navigation. A base index expression consists of terms that are linked to other terms by connectors. For example, "effective information retrieval" is a base index expression. So is "people in need of information". The two combined together form an index expression. For example, "effective information retrieval information AND people in need of information".

The power base expression is a lattice formed out of a full base expression at the top and an empty base expression at the bottom. This lattice (or lattice-like) structure is the basis of the hyperindex [Bruza, 1990]. Based on the vertex of focus in the lattice, the surrounding descriptors can represent enlargements (context extension) or refinements (context contraction) of the context represented by the focus. Thus, the reader can move across the lattice by refining or enlarging the current focus until a focus is found which is relevant to the information required.

Bruza’s measures to determine the effectiveness of index expressions in the hyperindex include:

a. ***Precision***: The ratio of relevant objects associated with the descriptor to the total number of objects associated with the descriptor.

b. ***Recall***: The ratio of the number of objects associated with the descriptor to the total number of relevant objects.

c. ***Exhaustivity***: The degree to which the contents of the objects are reflected in the index expressions.

d. ***Power***: The ratio of a descriptor’s specificity to its length.

e. ***Eliminability***: The ability to determine the irrelevance of a descriptor and stop the search.

f. ***Clarity***: The ability to grasp the intended meaning of the descriptor.

g. ***Predictability***: The ability to predict where relevant descriptors can be found in the index.

h. ***Collocation***: The extent to which the relevant index terms are near each other in the index.

Experiments and empirical studies are required to determine these retrieval measures for hypertext-based IR systems.

2.2. Structural Queries

Beeri and Kornatzky have suggested a logical query language that would allow structural queries over a hypertext network. First-order logic is too general since it ignores the particular characteristics of hypertext. Hence, there is a need for a structural query language to incorporate the notions of recursive and quantification constructs. The logic for the proposed query language is a mixture of propositional calculus (which has no predicates or variables) and quantifiers such as many, most, at least two, exactly five etc. [Beeri & Kornatzky, 1990]. The basic formulae of the logic are the propositions and assertions on attributes' values. Queries use specifiers to directly retrieve edges, paths, and cycles. The set of elements retrieved is collapsed into a hypertext network. The output of a query being a hypertext network, users can incrementally compose queries. Thus, the combination of specifiers, quantifiers, and the collapsing of query answers into a new hypertext network makes it possible to express structural queries proposed by Halasz. Facilities also exist in this query language to view portions of the retrieved network based on the specification of filters. Research is also underway to develop a visual hypertext query language.

GraphLog is a visual query language implemented on top of the Neptune hypertext system front-end to the Hypertext Abstract Machine (HAM) [Consens & Mendelzon, 1989]. It addresses the following issues raised by Halasz: Search and query mechanisms, augmentation of the basic node and link model, virtual structures, computation over the hypertext network, and versioning. Using GraphLog, queries are formulated by drawing graph patterns. These patterns are then searched for in the hypertext network to yield subgraphs.

GraphLog is highly expressive and it uses the notions of deductive database theory and descriptive complexity. The language is powerful enough to allow the specification and manipulation of arbitrary subsets of the network and supports the computation of aggregate functions on the subgraphs of the hypertext document. It can support dynamically defined structures as well as inference capabilities. GraphLog is an extension of the graph-based query language called G+. G+ was extended by adding the notions of negation, aggregation, and improved semantics to make it simple, yet powerful. It can express structural queries which cannot be expressed in conventional database languages such as relational algebra. For example, it can perform an arbitrarily long sequence of join operations for which there is no equivalent single relational algebra query. It was designed to avoid explicit use of logic formulae and recursion.

The concept of incremental construction of queries can also be found in HyperBase (Schutt & Streitz, 1990]. It provides two sets of retrieval operations, one set with operations that affect the whole hypertext database, another set of functions that operate only with respect to a sub-network. Streitz et al., are also working on the design of a Hypertext Query Language (HTQL).

2.3. Inference Networks

Indexes can also be considered as "precompiled links", providing immediate access to required information without navigating through the "document space" [Frisse & Cousins, 1989]. While nodes (containing information) can be considered part of the document space, indexes or index nodes can be treated as part of the "index space". In traditional full-text document retrieval systems, the document space and index space are essentially flat (they are together). Frisse and Cousins suggest the use of a hierarchical index space and a networked document space for information retrieval in hypertext systems. They have investigated the representation of index spaces as belief networks. Croft and Turtle have also proposed an IR model for hypertext systems based on plausible or non-deductive inference using Bayesian inference networks [Croft & Turtle, 1989]. Belief networks or Bayesian inference networks are directed, acyclic dependency graphs where nodes represent propositional variables and links or edges represent probabilistic relationships between the propositions. A hypertext system can be compared to such a network - the roots of the dependency graph are hypertext nodes; interior nodes and leaves represent concepts.

User's likes and dislikes are transmitted recursively from all nodes representing concepts (in the document space) to nodes in the index space. Based on this degree of belief, appropriate values are assigned to index space nodes. These changes are propagated throughout the index space using standard Bayesian techniques. If a proposition represented by a node **p** directly implies the proposition represented by node **q**, a directed graph is drawn from **p to q**. If-then rules in the Bayesian network are interpreted as conditional probabilities, that is, a rule **A --> B** is interpreted as a probability **P(B|A)**. Given a set of prior probabilities for the roots of the DAG the probability associated with the remaining nodes can be computed. Belief values of nodes of interest to a reader increase in value while nodes not of interest generally decrease in value. These probabilistic inference techniques applied to a hierarchical index space greatly enhances the information retrieval process in hypertext or networked document spaces.

Optimal retrieval effectiveness can be obtained by ranking nodes according to estimates of the probability that the query is true given a particular hypertext node. The ranking function is approximately equivalent to giving each node a score that is the sum of the weights of matching query terms, where the weights depend on the frequency of occurrence of a term in each hypertext node and in the entire collection of hypertext nodes. This simple retrieval model can be further extended by introducing dependencies that represent links between hypertext nodes. This means if hypertext node **j** is indexed by a particular concept and is linked to a hypertext node **k**, then there is some probability that hypertext node **k** should also be indexed by that concept.

However, the application of Bayesian techniques to a complex graph is NP-complete. It is also not clear whether adaptive IR systems based on Bayesian inference techniques converge on the appropriate set of index space nodes given imprecise information. Frisse and Cousins are of the opinion that the computational complexity of this approach needs more investigation. Some of the other restrictions include that the network topology cannot include directed cycles and that queries with evaluable predicates such as greater than, less than etc., cannot be handled.

Lucarella suggests a similar model for hypertext-based IR. While content nodes form the document network, there can be concept nodes forming the concept network (Figure 5.1) [Lucarella, 1990]. These concept nodes can serve as an index to the document network. The concept network is similar to the index space proposed by Frisse and Cousins. Links within the concept network establish associations between concepts. The type of the link can describe the nature of semantic association while a weight assigned to it can reflect the strength of the association. The resulting hypertext knowledge base (containing the concept network embedded within the document network) can be used for query analysis.

Based on a natural language query, the system will perform a search on the concept network to find the most pertinent sets of concepts. Concept recognition takes place by matching the various terms in the query and evaluating the matched expressions by applying similarity functions. Based on the degree of similarity between the concept and the matched item, a weight is associated to each concept. The search can be stopped based on the number of concept nodes examined and the number of top ranking concepts identified.

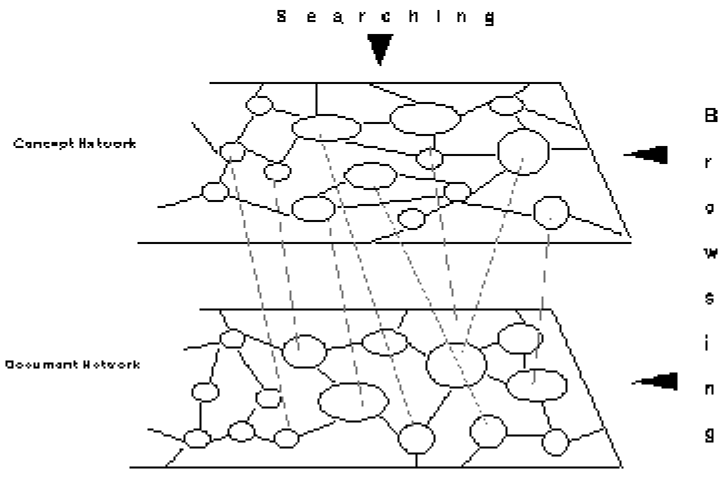

Figure 5.1. A Model for Hypertext-Based Information Retrieval [Lucarella, 1990].

Information is retrieved from the document network based on these retrieved concepts. Along with the exact set of documents, other sets of documents found to be semantically related to the topic of interest will be retrieved based on the weights assigned to the concepts. For example, a query to retrieve documents concerned with "expert systems" may also retrieve documents discussing "knowledge-based systems" since the two concepts are semantically related.

2.4. Cluster Hierarchies, Aggregates, and Exceptions

Researchers have suggested using the vector space model to organize a hypertext collection into clustered hierarchies [Crouch et al., 1989]. In this model, the content of each node or document is represented by a set of possibly weighted terms. Thus, each document can be represented by a term vector and the complete document collection can be represented by a vector space whose dimension is equal to the number of distinct terms to identify the documents in the collection. Similar or related documents are represented by similar multi-dimensional term vectors. Such a model facilitates clustering documents based on their similarity and ranking retrieved documents in decreasing order of their similarity to the query vector. Hence, the user can readily focus the search on those clusters that are likely to contain documents which are similar to the query. Comparisons are generally made between the query vector and the document vectors using one of the standard measures of similarity. Clustering is also helpful in locating neighboring nodes which discuss related topic(s). The user can incrementally refine the query vector to retrieve the desired document(s). An interactive browser incorporating the cluster hierarchy model was implemented by Crouch et al., on a Macintosh connected to a SUN network running the SMART Information Retrieval system. This interactive browser yielded a significant improvement over automatic cluster searches.

The object-oriented concept of abstraction (generalization/aggregation) will greatly benefit IR in hypertext systems [Botafogo & Shneiderman, 1991]. Abstraction is the concealment of all but relevant properties of an object or concept. Aggregation is the clustering of related objects to form a higher level object. For example, wheels, chassis, engine, body etc., can be aggregated together to form the composite object called "car". Generalization is the property of treating a set of similar objects as a generic object. For example, cars, trucks, buses etc., forms the generic object called "automobile". These two concepts of abstraction can be effectively used to simplify hypertext structures. A set of related nodes and the links between them can be treated as a semantic cluster (as opposed to the hierarchical cluster proposed by Crouch et al.) having the following properties:

a. They form a subgraph of the hypertext graph.

b. The compactness (the degree of interconnectedness of the hypertext)

of the subgraph is higher than the compactness of the whole graph.

Graph theoretical algorithms such as biconnected components and strongly connected components can be applied in the formation of such clusters or aggregates. A similar approach called Aggregation Clustering With Exceptions (ACE) has been proposed to identify aggregates or clusters and exceptions (those that do not fall into clusters) [Hara et al., 1991]. Such clustering mechanisms facilitate effective IR from hypertext systems.

3. Artificial Intelligence Techniques

A knowledge base and an inference engine built on top of the hypertext database can add "intelligence" to nodes and links. An interactive filter can be built which will consult with the user and call up the appropriate node. Since hypertext and AI have some common features (semantic dependencies between two pieces of information), research is required to merge the two in order to augment the intelligence of the user [Carlson, 1989]. Halasz reported that "Computation built into the hypermedia system is likely to be more efficient, especially when that computation involves extensive access to information in the network." [Halasz, 1988].

4. Information Retrieval and Browsing

Lucarella has written that whereas conventional IR techniques focus on "what to where" (we know what we want, but we wish to find out where in the database it is), hypertext browsers focus on "where to what" (we know where we are, but we want to know what is there) [Lucarella, 1990]. IR in a hypertext system can combine these two techniques to greatly enhance the process of finding relevant information. Hypertext browsing can supplement conventional IR by allowing users to discover retrieval cues that successively can be used for query formulation. Query facilities can supplement hypertext browsing by providing the user with a set of relevant nodes for browsing.

In a query language developed for HDM2, a query produces a list of references to items which can be units, composites, indexes, or guided tours. These can be used as a dynamic access structure (or virtual structure) from which further navigation can originate. The navigation space resulting from such a query or filter is called a hyperview. All subsequent requests (either queries or navigation commands) will be interpreted only within this restricted space [Garzotto et al., 1993].

5. Summary

While navigation or browsing is sufficient for small hypertext systems, more powerful information retrieval techniques become very important in large scale hypertext databases. Content queries can be used to retrieve the contents of nodes while structural queries can be used to retrieve subgraphs of the hypertext network that match a given pattern. Many researchers have investigated the possibilities of separating index information from contents thus forming an index space (or concept network) on top of a content space (or document network). These would not only facilitate IR but also accommodate dynamic linking and independent maintenance of the two networks.

Query languages are being extended to perform structural queries. These extensions include the notions of quantifiers, recursive operators, aggregation, and improved semantics. Research has also been carried out in the use of belief networks or Bayesian inference networks for hypertext-based IR. Some researchers have explored aggregating hypertext networks into semantic or hierarchical clusters. Very little work has been done in the area of merging Artificial Intelligence with hypertext. A combination of inference-based IR and knowledge-based hypertext could greatly facilitate browsing and searching. More research is required in the integration of querying techniques and browsing mechanisms. Experiments are required to measure the effectiveness of these IR techniques.

References

[Botafogo & Shneiderman, 1991]. Botafogo, Rodrigo and Shneiderman, Ben. Identifying Aggregates in Hypertext Structures, Proceedings of Hypertext ’91, ACM Press, 1991.

[Beeri & Kornatzky, 1990]. Beeri, Catriel and Kornatzky, Yoram. A Logical Query Language For Hypertext Systems, Proceedings of European Conference on Hypertext, ECHT ’90, 1990.

[Bruza, 1990]. Bruza, Peter D. Hyperindices: A Novel Aid For Searching in Hypermedia, Proceedings of European Conference on Hypertext, ECHT ’90, 1990.

[Carlson, 1989]. Carlson, Patricia Ann. Hypertext and Intelligent Interfaces for Text Retrieval, The Society of Text, MIT Press, 1989.

[Conklin, 1987]. Conklin, Jeff. Hypertext: An Introduction and Survey, IEEE Computer, September 1987.

[Consens & Mendelzon, 1989]. Consens, Mariano P. and Mendelzon, Alberto O. Expressing Structural Hypertext Queries in GraphLog, Proceedings of Hypertext ’89, ACM Press, 1989.

[Croft & Turtle, 1989]. Croft, W. Bruce, and Turtle, Howard. A Retrieval Model Incorporating Hypertext Links, Proceedings of Hypertext ’89, ACM Press, 1989.

[Croft et al., 1990]. Croft, W. Bruce, Belkin, Nicholas, Bruandet, Marie-France, Kuhlen, Rainer, Oren, Tim. Hypertext and Information Retrieval: What are the Fundamental Concepts, Panel Discussion, Proceedings of European Conference on Hypertext, ECHT ’90, 1990.

[Crouch et al., 1989]. Crouch, Donald B., Crouch, Carolyn J., and Andreas, Glenn. The Use of Cluster Hierarchies in Hypertext Information Retrieval, Proceedings of Hypertext ’89, ACM Press, 1989.

[Frisse & Cousins, 1989]. Frisse, Mark E. and Cousins, Steve B. Information Retrieval From Hypertext: Update on the Dynamic Medical Handbook Project, Proceedings of Hypertext ’89, ACM Press, 1989.

[Garzotto et al., 1993]. Garzotto, Franca, Mainetti, Luca, and Paolini, Paolo. Navigation Patterns in Hypermedia Data Bases, Proceedings of the Twenty-Sixth Hawaii International Conference on System Sciences, 1993.

[Halasz, 1988]. Halasz, Frank. Reflections on NoteCards : Seven Issues for the Next Generation of Hypermedia Systems, Communications of the ACM, July 1988.

[Hara et al., 1991]. Hara, Yoshinori, Keller, Arthur M., and Wiederhold, Gio. Implementing Hypertext Database Relationships through Aggregations and Exceptions, Proceedings of Hypertext ’91, ACM Press, 1991.

[Lucarella, 1990]. Lucarella, Dario. A Model For Hypertext-Based Information Retrieval, Proceedings of the European Conference on Hypertext (ECHT) ’90, 1990.

[Schutt & Streitz, 1990]. Schutt, Helge A., and Streitz, Norbert A. HyperBase: A Hypermedia Engine Based on a Relational Data Base Management System, Proceedings of ECHT ’90, 1990.

CHAPTER 6

INTEGRATION ISSUES

1. Introduction

Hypertext systems have not yet been accepted as fundamental tools for the augmentation of human intellect since most of them are "insular, monolithic packages that demand the user disown his or her present computing environment to use the functions of hypertext and hypermedia." [Meyrowitz, 1989]. Similar to the "cut and paste" paradigm in windowing environments, linking functionality must become an integral part of the computing environment. Application developers must be provided with tools to enable all applications to "link up" in a standard manner.

Most current hypertext systems are closed systems - material created in one system cannot be transferred or integrated with material created in another system because of proprietary document formats and storage mechanisms. Conversion programs are difficult to write since the formats are not disclosed by organizations [Fountain et al., 1990].

In order to make systems open and also integrate hypertext functionality into the desktop, researchers have been working on various hypertext models and interchange standards. This chapter explores these models and standards.

2. Models and Frameworks

2.1 Hypertext Abstract Machine (HAM)

One of the first approaches to a generic hypertext implementation model was the Hypertext Abstract Machine (HAM), "a general purpose, transaction-based, multi-user server for a hypertext storage system." [Campbell & Goodman, 1988]. HAM’s emphasis was on developing an appropriate storage model. It provided a general and flexible model that could be used in several, different hypertext applications. The Hypertext System Architecture based on HAM contains the following layers (See Figure 6.1):

User Interface: A window-based interactive environment for applications to communicate with users.

Application: The actual application which may or may not run on the same machine as the HAM.

Hypertext Abstract Machine: An engine which manages all information about the hypertext and communicates with the application through a byte stream protocol.

Host file system or storage system: A repository to store all the hypertext graphs or databases.

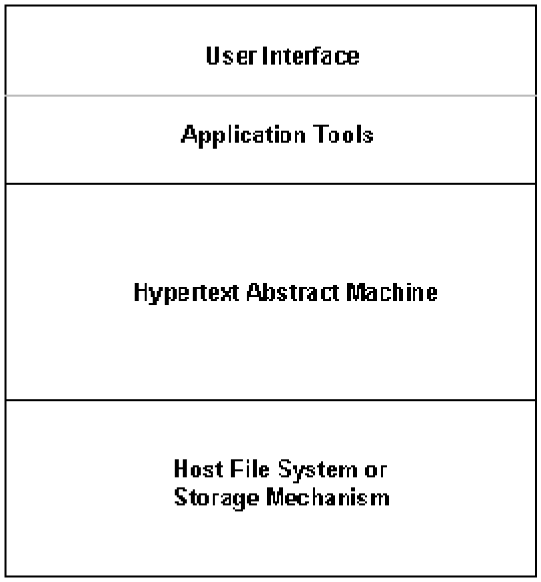

Figure 6.1 Hypertext Abstract Machine [Campbell & Goodman, 1988]

The HAM storage model consists of five major objects: graphs (networks of nodes and links containing one or more contexts), contexts (partitions of data within a graph), nodes, links, and attributes carrying semantics. The following operations could be performed on HAM objects: create, delete, destroy, change, get, filter, and special. The HAM architecture provided version control, filtering and data security. The HAM storage model has been successfully tested against systems such as Guide, Intermedia, and NoteCards.

2.2 Link Engine/Hypermedia Engine/Link Service

Developers of Intermedia were interested in developing a multiuser hypermedia framework whereby hypertext functionality could be handled at the system level - linking would be available for all participating applications [Haan et al., 1992]. They proposed "The IRIS Hypermedia Services" to provide an integrated desktop environment for hypermedia applications such as InterWord, InterDraw, InterVal, InterVideo, and InterPlay. These services contain the following components: Intermedia Layer, Link Client, and Link Server. These components are independent of both operating system and Graphical User Interface.

The documents themselves are stored as Unix files while the link and anchor data are stored in a DBMS. The Link Client, the Link Server, and the DBMS together form the Link Engine. The Intermedia Layer is responsible for all live data manipulation while the Link Engine is responsible for the storage and retrieval of link data. With such an approach Intermedia documents could be interchanged with KMS using the Dexter Interchange Format (described in Section 3.?).

According to Intermedia researchers, following are the requirements to make hypermedia an integrated part of the computing environment:

1. Integration of hypermedia into the desktop. The Link Engine must be integrated into the computing

environment just as the file system is today. A higher level toolkit or an application programmer interface (API) must be provided for application developers to issue calls for hypermedia support.

2. Hypermedia systems must provide multiple contexts or multiple webs in order to fully exploit hypertext linking across all applications.

3. Hypermedia applications must support filtering and incremental query construction.

4. Wide Area Hypermedia - Hypermedia functionality must be extended to support Wide Area Networks in addition to LANs.

5. Building an integrated hypermedia environment is made easier with object-oriented techniques. Also, the most logical DBMS to use for storing link and anchor data would be an Object-Oriented Data Base Management System.

In order to provide an integrated desktop with full hypermedia functionality, Bieber has proposed a system-wide hypermedia engine based on the notion of a generalized hypermedia using bridge laws (See Chapter 2, Section 5 - Dynamic Hypertext) [Bieber, 1993]. This engine would bind independent back-end applications such as Decision Support Systems, Expert Systems, Databases and front-ends (interface-oriented applications such as word processors, graphics packages) through message-passing mechanisms. Bridge laws map the objects defined in the back-end such as models, variables, calculations to objects in the front-end such as nodes, links, and link markers.

Bieber has suggested the following front-end and back-end requirements for system-level approaches to hypermedia integration or client/engine cooperation.

Front-end Requirements:

In order for the hypermedia engine to provide functionality such as management of link markers, comments, trails, overviews, filters, forward navigation and backtracking, the front-end should provide the following:

Tracking the location of objects such as link markers and providing their identifiers to the engine when a link marker is selected.

Front-end must request from the engine editing permissions for insertions, deletions, and modifications.

User interface must provide hypermedia prompts.

When the front-end saves a document with embedded hypermedia objects, the objects should also be saved.

Back-end Requirements:

The hypermedia engine would provide functionality such as linking, annotation, backtracking, filtering, and overviews on behalf of the back-end. The back-end should provide the following functionality:

Provide specific information about its structure and its applications' documents.

Bridge laws must be written by developers. This could be done through a bridge law editor instead of writing predicate logic.

Back-ends should provide control information and interpretive mechanisms along with the objects that are sent through messages. For example, objects that have to serve as link markers could be tagged.

Back-ends should support hypermedia engine commands same as the front-end (command lists and context sensitive information).

Back-end should incorporate a standard document interchange standard such as ODA or SGML.

Similar to Intermedia's Link Engine and Bieber's Hypermedia Engine, Sun's Link Service offers an extensible protocol to create and maintain relationships between autonomous front-end applications [Pearl, 1989]. Similar to the approaches seen earlier, editing and storing of data objects is managed by independent applications which also provide some amount of front-end operations on links. The Link Service stores only the representations of the nodes rather than the nodes themselves. Thus, the definition and granularity of nodes are left to the individual applications. Also, the storage of node data is independent of the storage of link data.

The Link Service makes it easier for applications to add hypertext functionality by providing a simple protocol, a shared back-end or link server, a library, and utilities to manage the link database (See Figure 6.2). Applications communicate with the link server through the Link Service protocol. This service allows independent applications to integrate linking mechanisms into their standard functionality and become part of an extensible and open hypertext system. Existing text and graphics editors can be integrated into such a framework without any modifications. Due to the separation of node and link data, the Link Service does not provide version control, node content editors, concurrent multi-user access, or other forms of data integration.

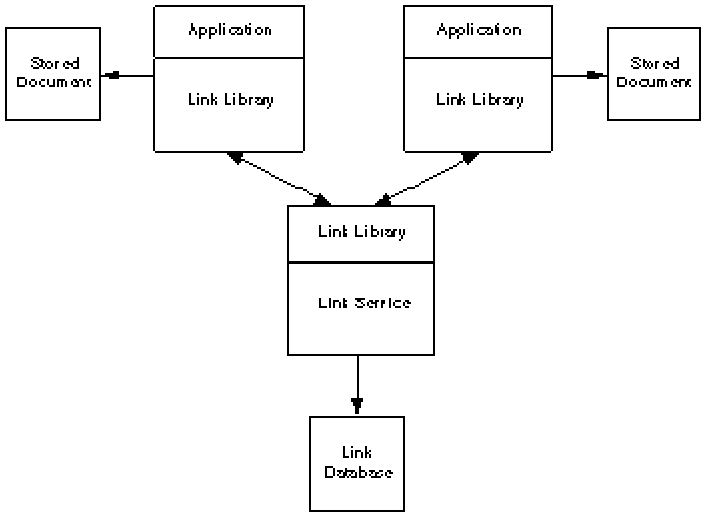

Figure 6.2 Link Service - An Architecture for Open Hypertext [Pearl, 1989].

Some of the issues involved in developing such an open hypertext system include the following [Pearl, 1989]:

The User Interface for the creation and management of links should be consistent with the editors provided by the individual applications.

Since the Link Service and the applications are separate processes decisions must be made about sharing/dividing the responsibility for exception handling and user dialogs.

The Link Service should detect and remove dangling links, either implicitly or explicitly. Implicit removal can happen when a user tries to follow a link from its valid end to its invalid end by suggesting to the user to remove the link. Explicit removal can happen, through a link garbage collection mechanism, by tracking links, validating nodes and removing invalid links.

While versioning of data objects can be left to the individual applications, the Link Service must still handle the versioning of links. However, the consistency of a versioned hypertext cannot be guaranteed if nodes are versioned separately from links.

Unstructured documents such as ASCII files cannot be handled elegantly since they are not uniquely indexed nor do they carry semantics.

The issue of traversing links across networks and locating objects located at remote sites is very important due to performance and cost factors. Also, decisions have to be made about where on the network should the Link Service process be located. Another related issue is the invocation of an application which may not be currently running although the user is following a link to a node managed by that application.

2.3 Hypermedia Toolkit

The toolkit approach mentioned by Haan et al., has been attempted by Puttress and Guimaraes. They have proposed a toolkit that could be used by application developers to add hypermedia functionality to their existing toolkit, independent of specific applications or environment [Puttress & Guimaraes, 1990]. The hypermedia toolkit architecture is similar to other multi-tiered architectures (See Figure 6.3). The layers are: Application Software, Hypermedia Toolkit Layer, Storage System, and Representation System. The hypermedia toolkit consists of the following three components:

a) ***Storage System Interface*** (also called Eggs): This interface consists of a set of C++ classes, providing a hypermedia structure to the stored application data. It provides the mapping between the application above and the storage system below. Thus, the storage system can be modified or changed without modifying the application. Similar to the HAM approach, the data model is made of graphs, contexts, nodes, links, attributes, and symbols. This interface does not interpret node data - it is just considered as a stream of bytes with no structure or meaning. It provides version control and concurrency control mechanisms. There is finer transaction management under the control of the application.

b) ***Application Interface:*** This interface is composed of data objects that communicate with the application above.

c) ***Representation System Interface:*** This interface is responsible for the presentation of views using user interface toolkits, independent of the display platform. The Application Interface and the Representation Interface are made of a set of C++ classes, together called Hypermedia Object-oriented Toolkit (HOT). HOT provides the abstractions required for hypermedia applications while encapsulating the details of the storage and representation systems. HOT consists of Data classes that include: HGraph, HContext, HNode, and HLink. It also consists of View classes for each of the Data classes: HGraphView, HContextView, HNodeView, HLinkView and HFrame.

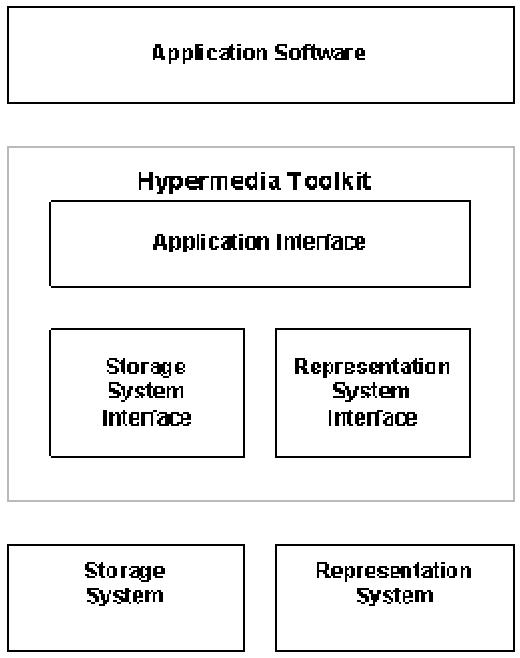

Figure 6.3 Hypermedia Toolkit Architecture [Puttress & Guimaraes, 1990].

Puttress and Guimaraes report that this architecture will be extended to support multi-user environments, to provide effective means of sharing and communication between users of hypermedia applications, and exploring means to the development of collaborative hypermedia [Puttress & Guimaraes, 1990].

2.4 HDM

The Hypermedia Design Model (HDM) is a hypertext model being developed as part of the HYTEA project by an European Consortium [Garzotto et al., 1991]. The basic features of HDM include the representation of hypertext applications through primitives: types entities composed of hierarchies of components; different perspectives for each component; units corresponding to component-perspective pairs; bodies representing the actual contents of units; structural links relating components belonging to the same entity; application links relating components belonging to different entities; and browsing semantics determining the visualization and dynamic properties of the application. These primitives are similar to objects defined in HAM.

The HDM is concerned with authoring-in-the-large or the definition of the topology of the hypertext network. It does not deal with authoring-in-the-small or filling in the contents of nodes and their presentation. This is because, Garzotto et al., believe that systematic and rational structural decisions about the hypertext should be made before the actual hypertext is written so that a coherent and expressive application can be developed from the very beginning instead of being added later. These HDM design specifications can be translated automatically into a lower-level node and link specification resulting in the actual implementation of the topology. HDM is still evolving and requires experimentation with a large number of applications. It is a step to induce a methodology for hypertext design based on a top-down, model-based approach.

2.5 Dexter Hypertext Reference Model

The Dexter Hypertext Reference Model captures the important abstractions found in a wide range of existing and future hypertext systems [Halasz & Schwartz, 1990]. The goal of the model is to provide a systematic basis for comparing systems and to develop interchange and interoperability standards. The Dexter model divides a hypertext system into three layers (See Figure 6.4):

a. Runtime Layer

This layer deals with the presentation of hypertext and the dynamics of user interaction. Since it is too broad and diverse to be developed into a generic model, the Dexter model does not go into the details of the presentation mechanism. However, presentation mechanisms can be specified containing information about how a component/network is to be presented to the user. These presentation specifications provide an interface between the runtime layer and the storage layer.

b. Storage Layer

This is the main focus of the Dexter model. It models a database that is composed of a hierarchy of data-containing components which are interconnected by relational links. Components have unique identifiers and links can be identified by a set of two or more component identifiers. Components correspond to the general notion of nodes and can contain text, graphics, images, audio, video etc. The components are treated as generic containers of data and the model does not specify any structure within the containers. Thus, the storage layer does not differentiate between text components and graphics components. It focuses mainly on the mechanism by which components and links are tied together to form hypertext networks.

c. Within Component Layer

This layer is concerned with the contents and structure within components of the hypertext network. Since the range of possible content/structure that can be included in a component is open-ended, the Dexter model treats this layer as being outside its scope. The assumption is that document structure models such as ODA, SGML, IGES etc., will be used in conjunction with this model to capture content/structure. However, a critical interface between the storage layer and the within-component layer called anchoring discusses the mechanism of addressing locations or items within the content of an individual component. Anchors can identified by a unique anchor identifier.

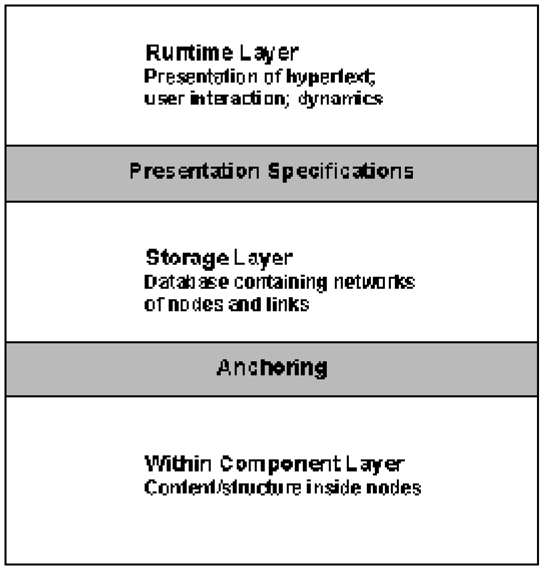
Runtime Layer
Presentation of hypertext; user interaction; dynamics
Presentation Specifications
Storage Layer
Database containing networks of nodes and links
Anchoring
Within Component Layer
Content/structure inside nodes

Figure 6.4 Dexter Hypertext Reference Model [Halasz & Schwartz, 1990].

Halasz and Schwartz claimed that no existing systems support all the mechanisms discussed in the Dexter model. However, existing systems are yet to be fully compared to this model. The model has been used in developing the Dexter Interchange Format, a hypertext interchange standard.

2.6 Trellis Hypertext Reference Model

The Trellis Hypertext Reference Model or "r-model" looks at hypertext as different levels of abstraction [Furuta & Stotts, 1990]. In general, hypertext can be divided into:

a) Abstract Level: This layer is made of abstractly defined independent components that are connected together in some fashion. It does not describe the details of presentation.

b) Concrete Level: Concrete representations in which the characteristics of the hypertext's physical display have been established. That is, the contents of each of the windows is specified but not laid out.

c) Visible Level: This layer is responsible for the layout and presentation of the hypertext network on a physical display.

The representations in the abstract level are at the greatest level of abstraction while those in the visible level are the lowest level. Whereas the abstract level can be standardized using document exchange protocols such as SGML, the visible level can be standardized using X Windows.

While some researchers believe that hypertext can be standardized, there are others who contend that the current notion of hypertext is not mature enough to be standardized. There is the third group which believes that currently identified and well-established features of hypertext may be standardized [ECHT ’90 Panel Discussion, 1990]. Hardt-Kornacki et al., feel that "the components of hypermedia that are ready for standardization are not necessarily hypermedia-specific and the hypermedia aspects of these systems are not yet ready for standardization." [Hardt-Kornacki et al., 1990]. They contend that objects which have data and structure need rich and flexible standard representations in matters of exchange and authoring. However, since the same objects might be involved in non-hypermedia applications, they question the prudence of hypermedia-based object presentation standards. However, it is not clear if they mean that models such as Dexter and Trellis are not yet required. Also, standardizing the user interface for hypermedia systems is premature and naive. They add that it may be beneficial to standardize a generic set of tools that can be used to build various components of hypertext. They contend that none of the supporting infrastructure in a hypermedia environment such as display devices, user interfaces, file systems, broadband communications etc., are standardized.

However, such layered approaches (back-end, engine, and front-end) require significant efforts from software developers in writing exchange protocols between the layers. If these protocols are application dependent, they cannot serve as standard approaches. Hence, Isakowitz feels that alternatives to the layered architecture approach should also be explored [Isakowitz, 1993]. Mechanisms have to be devised to automate this layer-to-layer communication so that they are independent of applications.

2.7 General Hypertext Framework

Whereas most models have focused on design metaphors and implementation abstractions, very little work has been in the area of a general framework for hypertext functionality. Rao and Turoff observed that "Hypertext should be treated as a general purpose tool with approaches to handling nodes, links, and retrieval, that fits within the context of any application and conveys common meanings to users. To accomplish this, we need a comprehensive framework for hypertext based on a cognitive model that allows for the representation of the complete range of human intellectual abilities." [Rao and Turoff, 1990]. They proposed such a framework based on Guilford’s Structure of the Intellect Model (See Figure 6.5). They contend that hypertext systems tend to suffer from a lack of coherence due to ambiguity in meanings assigned to nodes and links. This framework classified nodes into six different semantic types - detail, collection, proposition, summary, issue, and observation. Links can be categorized into major types - Convergent links and Divergent Links. Convergent links can be classified into specification, membership, association, path, alternative and inference links. These links help in focusing or narrowing the pattern of relationships between ideas. Divergent links are classified into elaboration, opposition, tentative, branch, lateral, and extrapolation links. These links expand or broaden relationships between ideas.

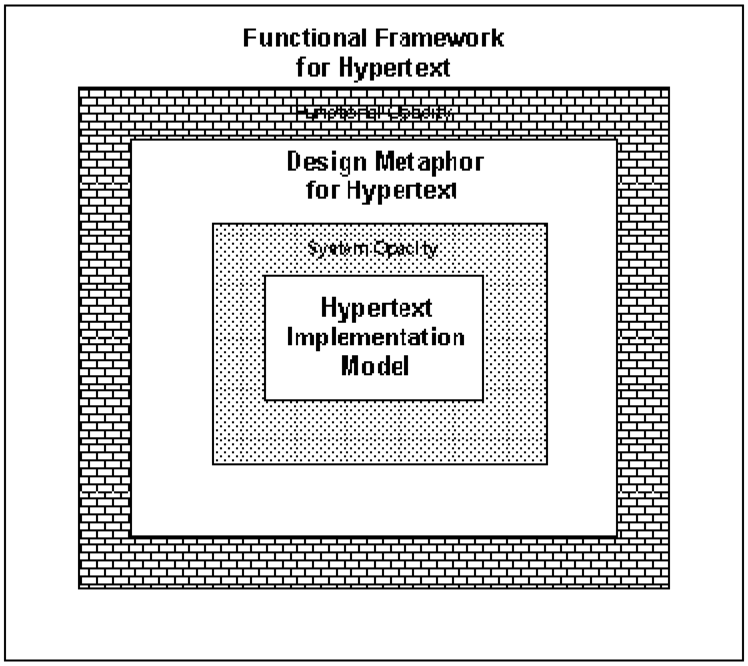

Figure 6.5 General Framework for Hypertext Functionality [Rao & Turoff, 1990].

Rao and Turoff believe that such a comprehensive framework would help designers develop better interface metaphors and implementation models for hypertext systems. They concluded that sixteen different hypertext systems which they reviewed fall under this framework (in their own limited ways) and that their semantic morphology could be extended to all future systems. Such a taxonomy would also help collaborative hypertext where members of a group could contribute adequately and understand each other's judgements in carrying out the group objective. A first step towards the implementation of a hypertext system based on such a framework is to develop an appropriate design metaphor/user interface that would reduce functional opacity (mismatch between the framework and the metaphor) and system opacity (mismatch between the metaphor and the implementation model). That is, the region called design metaphor in Figure 6.5 could be expanded on all four sides thus "squeezing out" the shaded regions named functional opacity and system opacity.

3. Interchange Standards

Unlike linear documents which are static, generic, and structured, hypertext documents are unstructured and can be dynamic. Hence, current structured document standards are not sufficient to represent hypertext networks. A tree based hierarchy is relevant but NOT sufficient for hypertext. There should be a hierarchical framework with a system of typed links to cover the cross-references of structured documents and the links of hypertext. The current forms of ODA and SGML are not sufficient enough for the representation and exchange) of hypertext. These need extensions to provide a proper typed-link mechanism. SGML does not specify layout or presentation information (which is important for hypertext) or how to handle images and graphics. ODA does address these issues but it is not sufficient.

At the same time, a single standard may not be enough due to the diversity of usage of hypertext applications - large volume hypertext systems are different from highly interactive systems. While the former require highly efficient search capabilities (and standards), the latter require better individualized responses and navigational tools. Hence, these two may require different standards.

In this section, we discuss specific limitations of SGML and ODA and research efforts to extend them to handle hypertext [Newcomb et al., 1991]. Some researchers are also interested in combining the best of ODA and SGML and extending them to form a comprehensive hypermedia standard.

3.1 Limitations of SGML

1. SGML allows cross-referencing within the same document. This can be done by assigning unique identifiers to elements that need to be referred to elsewhere. However, the uniqueness of the identifier (and hence, the element) is applicable only within the current local document. Hence, only elements within the same document (and only those having unique identifiers) can be linked. Therefore, this mechanism can only be used in a hypertext document to refer to elements within the same document and not other documents.

2. SGML cannot support time dependent data such as audio and video and also graphics and images. Rendering of events is not possible in SGML, that is, displaying a map of NY and a link that zooms into Manhattan.

3.2 Limitations of ODA and possible modifications

ODA, a standard for the storage and interchange of multimedia documents, deals with both logical structure and layout structure or presentation (unlike SGML). ODA currently includes graphics and images and extensions are being considered to handle audio, video, and hypertext [Cole & Brown,1990].

a. Separation of logical structure and layout structure

Though ODA supports both logical structure and layout structure, they are not completely separated. In order to change the style of a document the logical structure must be edited since the layout process uses the logical structure, the generic structures and the content architectures to create the specific layout. This limitation can be eliminated by carrying over the SGML mechanism of applying different set of layout and presentation styles (or style sheets) for different views of the same logical document.

b. Comprehensive attribute inheritance

The ODA mechanism for inheriting layout attributes (such as placement of blocks of contents within pages and rectangular areas called frames) and presentation attributes (such as character sets and the placement of items within blocks) is not sufficient. If an attribute value is not specified for the object or its class, then the value can only be inherited according to the object’s position in the tree and not according to its class (chapter, list etc.).

Attribute inheritance can be achieved by adding a facility called "style tables" which will enable the style inherited by an object (and hence its format) to depend both on its class and is position in the document. This will be very valuable for hypertext in order to distinguish between objects of the same type that have different status (such as open and close buttons). It can also be extended to specify changes of state (for example, when selecting a hotspot) by changing the style table.

c. Links

ODA does not have the ability to specify the purpose of a link and also how the layout process can express that purpose. This can be accomplished by having classes for links (just as there are classes for logical objects). The class of the link will determine how and where in the document the link can be used. Thus, the representation of the link will depend on both the class and its position in the document.

d. Selective and multiple presentation

ODA does not have the ability to suppress the appearance of a logical object (or contents) during the layout process nor the ability to present the object many times. Such a feature will be of great help in a hypertext document where a reviewer’s comments can be suppressed from appearing in a printout or different versions of the same basic document can be produced for various purposes. This can also be accomplished by the usage of style tables suggested earlier.

e. Complete interactivity

The ODA layout process is sequential and page based and hence does not provide complete interactivity. It does not support online editing capabilities such as the ability to scroll through a document, the ability to display selected items (outlining facility), the ability to popup additional information on demand (such as footnotes, glossaries etc.), the ability to "fold" documents revealing hidden sections only on request, the ability to follow links automatically.

Complete interactivity would require extensions. Outlining can be done by having style tables that select objects by class and required level. Popup displays can be arranged by changing to a different style table and returning to the original table after the popup information has been displayed. Similarly, folding can be achieved indirectly through popups and popdowns. Link traversal can be done by replacing the current object with the target object or displaying the target object as a temporary popup item. A style table can be used to specify whether or not to display the linked object.

3.3 HyTime

The Hypermedia/Time-based Structuring Language or HyTime is and International Standard for representing hypertext links, and synchronization of static and time-based information contained in multiple conventional and multimedia documents and information objects [SIGLINK, 1992]. It addresses the limitations of SGML [Newcomb et al., 1991]. HyTime supports cross-referencing facilities to uniquely identified elements in external documents. It also extends SGML’s reference capability to accommodate elements with no unique identifiers in the same document. It provides pointers or location addressing schemes that contain the necessary information in order to locate cross-referenced data. It is independent of data content notations, link types, processing, presentation, and semantics. HyTime supports addressing by name, by position in the document, and by semantic construct. Links can be established to documents that conform to HyTime as well as those that do not.

HyTime allows all kinds of multimedia and hypertext technologies (whether proprietary or not) to be combined in any information product. It addresses only the issue of interchange of hypermedia information and not the standardization of presentation (same as SGML), user interfaces, query languages etc. Objects in a HyTime hypertext document can include formatted and unformatted documents, audio and video segments, still images, animations, and graphics.

HyTime is an SGML application conforming to ISO 8879. It provides the notion of "Architectural Form" to SGML. An architectural form is a syntax template around which a document author can build semantic constructs for linking and coordinate space addressing. It is highly flexible and extensible. The interchange format can be defined in Abstract Syntax Notation 1 (ISO 8824) and can be encoded according to the basic encoding rules of ISO 8825 for interchange using protocols conforming to the OSI model. The full set of HyTime functionality supports "integrated open hypermedia", the "bibliographic model" of hyperlinking that allows links to anything, anywhere, anytime.

HyTime is intended for use as the infrastructure of platform-independent information exchange for hypermedia and synchronized and non-synchronized multimedia applications. Application developers will use HyTime constructs to design their information structures and objects and the HyTime language to represent them for interchange [SIGLINK, 1992].

3.4 MHEG

CCITT has proposed the future international standard for multimedia and hypermedia information objects, also known as the MHEG Standard. "The scope of the MHEG standard is to define the representation and encoding of multimedia and hypermedia information objects that will be interchanged as a whole within or across applications or services, by any means of interchange including storage devices, telecommunications or broadcast networks." [CCITT, 1992]. The initial objectives of the MHEG standard include meeting the following requirements:

Provide abstractions for real-time presentation including multimedia synchronization and interactivity.

Provide abstractions for real-time interchange with minimal buffering using normal speed data communications.

Provide abstractions for direct manipulation of information without any additional processing.

Provide linking facilities between elements of composite multimedia objects.

The main MHEG classes include: Content Class, Selection Class and Modification Class, Link Class, Script Class, Composite Class, and Description Class. The objects play a federating role, enabling different applications to share the basic information resource. These objects can be encoded using ASN.1 or SGML and will provide a common base for other CCITT recommendations, ISO and other standards, user defined architectures and applications.

4. Summary

Hypermedia systems have been closed systems with proprietary storage mechanisms and very little or no interoperability. A number of layered architectures, models or engines, and frameworks have been proposed and developed by researchers in an effort to make hypertext systems more generic and integrated into the desktop environment. Application development toolkits have been developed to assist programmers in adding hypertext functionality to existing systems. In order to make hypertext systems fully open and integrated, the following issues must be addressed: interoperability, programmability, node and link typing, distributed linking, concurrency control for multi-user access in a shared environment, maintaining public and private links, operating systems support, networking, bridge laws, linking protocols, multimedia support, operating systems support, user interface consistency, and version control [Malcolm et al., 1991]. Most of these requirements can be addressed using object-oriented techniques [Lange, 1993].

In order to make hypertext systems fully portable, existing document standards such as ODA and SGML must be extended to support unstructured documents and linking. International standards such as HyTime and MHEG are emerging to support hypertext functionality and multimedia information in applications. Only when hypertext functionality becomes an integral part of our computing environment will knowledge workers accept and incorporate hypertext into their daily work process.

References

[Bieber, 1993]. Bieber, Michael. Providing Information Systems with Full Hypermedia Functionality, Proceedings of the Twenty-Sixth Hawaii International Conference on System Sciences, 1993.

[CCITT, 1992]. CCITT Press Release on MHEG, December 1992.

[Campbell & Goodman, 1988]. Campbell, Brad, and Goodman, Joseph M. HAM: A General Purpose Hypertext Abstract Machine, Communications of the ACM, July 1988.

[Cole and Brown, 1990]. Cole, Fred and Brown, Heather. Standards : What can Hypertext learn from paper documents ?, Proceedings of the Hypertext Standardization Workshop by NIST, Jan 1990.

[ECHT ’90 Panel Discussion, 1990]. Panel Discussion. Hypertext and Electronic Publishing, Proceedings of European Conference on Hypertext, November 1990.

[Fountain et al., 1990]. Fountain, Andrew M., Hall, Wendy, Heath, Ian and Davis, Hugh C. MICROCOSM: An Open Model For Hypermedia With Dynamic Linking, Proceedings of ECHT ’90, 1990.

[Furuta & Stotts, 1990]. Furuta, Richard and Stotts, P. David. The Trellis Hypertext Reference Model, Proceedings of the Hypertext Standardization Workshop by National Institute of Science and Technology (NIST), January 1990.

[Garzotto et al., 1991]. Garzotto, Franca, Paolini, Paolo, and, Schwabe, Daniel. HDM - A Model for the Design of Hypertext Applications, Proceedings of Hypertext ’91, ACM Press,December 1991.

[Haan et al., 1992]. Haan, Bernard J., Kahn, Paul, Riley, Victor A., Coombs, James H., and Meyrowitz, Norman K. IRIS Hypermedia Services, Communications of the ACM, January 1992.

[Halasz & Schwartz, 1990]. Halasz, Frank and Schwartz, Mayer. The Dexter Hypertext Reference Model, Proceedings of the Hypertext Standardization Workshop by National Institute of Science and Technology (NIST), January 1990.

[Isakowitz, 1993]. Isakowitz, Tomas. Hypermedia, Information Systems, and Organizations: A Research Agenda, Proceedings of the Twenty-Sixth Hawaii International Conference on System Sciences, January 1993.

[Lange, 1993]. Lange, Danny. Object-Oriented Hypermodeling of Hypertext Supported Information Systems, Proceedings of the Twenty-Sixth Hawaii International Conference on System Sciences, January 1993.

[Malcolm et al., 1991]. Malcolm, Kathryn C., Poltrock, Steven E., and Schuler, Douglas. Industrial Strength Hypermedia: Requirements for a Large Engineering Enterprise, Proceedings of Hypertext '91, ACM Press, December 1991.

[Meyrowitz, 1989]. Meyrowitz, Norman. The Missing Link: Why We're All Doing Hypertext Wrong, The Society of Text, MIT Press, 1989.

[Newcomb et al., 1991]. Newcomb, Steven, ????, and Newcomb, Victoria. The HyTime Hypermedia/Time-based Document Structuring Language, Communications of the ACM, November 1991.

[Pearl, 1989]. Pearl, Amy. Sun's Link Service: A Protocol for Open Linking, Proceedings of Hypertext '89, ACM Press, 1989.

[Puttress & Guimaraes, 1990]. Puttress, J.J., and Guimaraes, N.M. The Toolkit Approach to Hypermedia, Proceedings of the European Conference on Hypertext '90, 1990.

[Rao & Turoff, 1990]. Rao, Usha & Turoff, Murray. Hypertext Functionality: A Theoretical Framework, International Journal of Human-Computer Interaction, 1990.

[SIGLINK, 1992]. SIGLINK Newsletter. Excerpts on the HyTime International Standard, March 1992.

CHAPTER 7

APPLICATIONS

1. Introduction

Many existing information management systems that try to capture relationships between real world entities are not able to express them completely due to limitations in their structure and information retrieval techniques. The hypermedia paradigm provides a new and improved approach to developing complex information management systems in order to increase their effectiveness and usage. In this chapter, we review some of the applications based on the concept of hypermedia.

2. Encyclopedias, Dictionaries, Manuals, Handbooks, and Online Documents

A hypertext form of a voluminous dictionary such as the Oxford English Dictionary (OED) will not only help find meanings for words but also help us manage our daily knowledge work better [Raymond & Tompa, 1988]. In addition to browsing, complex queries can be built and intelligent searches can be performed. That is, users typically refer to dictionaries as part of more extended tasks. Improved navigation and IR mechanisms will greatly assist the information gathering process. Since much of the OED is made of quotations, links can be directly established to the source of the quotation or to other texts.

The field of medicine requires immediate evaluation and treatment of most ailments. Many a times, physicians have to recall general approaches to diagnosis and treatment of a particular ailment based on their previous experience with other patients. They have to retrieve previous medical records of patients, literature on the subject published in a journal, notes from another colleague etc. Such associations of material from diverse sources can be made easily accessible by using hypertext based information systems. Washington University developed the Dynamic Medical Handbook to provide immediate and easy access to medical information to medical professionals [Frisse, 1988]. Chapters from the Manual of Medical Therapeutics were converted to hypertext format. The handbook combined traditional IR mechanisms with browsing techniques to improve information retrieval. Such dynamic medical books, also called "hyperbooks" are widely used in medical schools. These hyperbooks can be integrated with medical media such as X-rays, CT images, MRI images, Charts, and Graphs into a medical "hyperlibrary". Such a system can help collaborative work in medicine where physicians, radiologists, and surgeons can diagnose and suggest treatments over a distributed hypermedia network with telepointing, annotation etc.

Users tend not to read printed manuals since finding the relevant information is cumbersome. Hypertext systems are the obvious choice for providing online documentation since they enable easy and flexible ways of accessing only relevant information. Online documentation can be combined with online help, tutorials, and error recovery. For example, an error message can be linked to some help text which can elaborate on the problem - why it occurred and how it can be remedied. Nielsen calls such online assistance as integrated user assistance [Nielsen, 1990].

Incorporating hypermedia in documentation allows document designers and users to create an unlimited number of independent contexts or perspectives over the same hypertext network, each constructed for a different category of users [Maurer & Tomek, 1990]. Such tailorable tutorials or views can cater to novices as well as experts using the same corpus of material. This eliminates duplication of the material for different audiences.

3. Learning Systems, Museum Exhibits, and Interactive Kiosks

Hypertext paradigm is highly suitable for learning systems in order to encourage learning by exploration. Epics and classics can also be represented in hypertext form. The Shakespeare Project was started in order to exploit multimedia and hypertext technology providing educators a radically new way to teach not only Shakespeare and theater, but also disciplines such as psychology, sociology, and communications which depend on the observation of complex, visually dense, and hard-to-record events [Friedlander, 1988].

Various applications in the Intermedia environment have together been used to create educational material for courses in English and Cell Biology [Yankelovich et al., 1988]. These interactive courses also encouraged students to submit term papers using hypermedia tools by referring to each others' papers. These experiments illustrated some interesting aspects of group dynamics involved in a collaborative class environment.

Museum goers gather information by exploration and hypertext systems can be used to describe the various exhibits. Hypertext-based museum information systems can provide visitors with a greater degree of involvement [Shneiderman & Kearsley, 1989]. The HyperTies system has been used at several museums, including the Smithsonian Institution, to describe museum exhibits on the Holocaust and "King Herod's Dream". Interactive kiosks can be used as tourist guides since tourists would like to read only parts of the information available about a city or country. Glasgow Online is another popular hypertext tourist guide that provides a subject-oriented overview such as city profile, accommodation, maps, places of interest, shopping centers etc. Selecting one of these topics will take the user through a trail of items related to the topic. The tourist can be looking at the city profile and be interested in a hotel in a certain price range. He or she can then search for accommodation fitting the requirement which would yield a list of hotels. The hotel of interest can then be located in a map of the region.

4. Idea Processing

Hypertext systems can be used for idea processing and brainstorming since both are made of a large number of disparate chunks of information put together by associations. Conferencing systems support idea processing and are suitable for hypertext features. Hypertext systems can also be used in journalism since it involves putting together various news items and stories. In fact, research or literature review such as this paper, will greatly benefit through hypertext authoring tools instead of word processors. We should be able to scan in the original documents (copyright laws permitting) used as references. When a reference is made to a document, a reference item should be created and the user should have the option to look at the original document on the screen.

NoteCards supports such collaborative idea processing where users in a group can create nodes, add links to nodes created by others, and add annotations to other nodes. It provides a rich and extensible set of tools to capture, represent, link, manage, and communicate ideas [Halasz, 1988]. Users can transform informal and unstructured ideas into formal analyses and structural representations. In addition to end-user tools, NoteCards provides a set of well-defined methods and protocols for programmatically manipulating the hypertext network. Examples of applications in which NoteCards has been used are researching and writing complex legal briefs, analyzing arguments presented in scientific and public policy articles and managing general personal and project information for small group research projects. It has also been used for a number of interactive, non-linear, multimedia reference manuals.

SEPIA (Structured Elicitation and Processing of Ideas for Authoring), is an active, knowledge-based authoring and idea processing tool for creating and managing hypertext documents [Streitz et al., 1989]. The main principle behind this system is the idea of cognitive compatibility - the system provides an environment to the author consisting of properties inherent to different cognitive activities and structures of writing. The system is task-oriented and through different representations supports the easy mapping of internal structures to external structures and vice versa. The system provides a variety of "activity spaces" which differ both in structure and functionality. The "content space" provides facilities to collect information, generate, and structure ideas for the content domain. The "planning space" provides support to the author in setting up an agenda, coordinating and directing all sub-activities, and monitoring and revising plans and goals. The "argumentation space" serves as the medium for generating, ordering, and relating arguments for specific issues, addressing one issue at a time. The "rhetorical space" allows structuring the outline of the final document, the rhetorical organization of positions and arguments for each sub-issue and the development of a coherent document.

The writing process is not straightforward but a criss-cross of interactions between these activity spaces. Information flows from the planning space to the other three phases: Issues specified in the planning space set topics for the content space, direct the structuring of positions and arguments for the argumentation space which are then transformed into an outline in the rhetorical space. On the other hand, information also flows directly from these spaces back to the planning space. This helps in refining the planning space also called "opportunistic planning".

5. Collaborative Work and Computer Mediated Communications

Hypermedia is highly appropriate for collaborative work. In addition to being used as a tool for idea processing, NoteCards has been used for collaborative writing. Since this activity can be highly coupled and unstructured, hypermedia is the most appropriate technology to represent both the work and the discussion about the work (meta-discussion) and about sharing the medium [Irish & Trigg, 1990]. Also, the rationale for decisions made during the writing activity can be preserved for historical purposes. NoteCards provided the environment for people to capture, structure, compare, and manage large amounts of loosely structured textual, graphical, and numerical information. In order to support the social interactions inherent in collaborative work and shared access to networks of information, NoteCards required multi-user access and concurrency control mechanisms, and version control. Irish and Trigg state that hypermedia is a good medium for supporting meta-discussions for the following reasons:

Meta-discussion often needs to directly refer to portions of the actual discussion.

A record of the meta-discussion preserved in context can be valuable for historical and chronological access.

From a Computer Mediated Communications (CMC) point of view, hypertext can be "viewed as a mechanism to enable an individual to impose a cognitive viewpoint on a collaborative database, and to facilitate group agreement on a shared understanding of the resulting semantics." [Turoff et al., 1991]. In a group project, each member of the team is working towards a sub-goal which forms part of a larger objective for the organization. Such an environment requires tracking, linking, and reorganization of information related to various sub-goals and the main goal itself. It can be argued that every CMC system capable of mail, conferences, comments, replies, and notifications can be viewed as a variation of the hypertext concept. For example, in EIES2, the facilities to reply to a mail item or add an attachment which form branches to the original piece of material can be considered a hypertext feature. The ability to cross-reference through keywords, and associated comments can be considered another hypertext feature.

Turoff et al., extend the general framework for hypertext functionality (covered in Chapter 6), to include a general morphology of terms for node and link types. Such a general approach allows users of a system to freely assign their own semantics to objects and relationships. The semantic model for hypertext-based collaborative systems should support the full range of human intellectual abilities. That is, such a system should allow any group member to define and describe objects (nodes) and relationships (links) between objects incorporating the full range of possible meanings for those nodes and links. Collaborative hypertext should also provide information about the composition or evolution process, the way the network is dynamically changing. This can be achieved by creating a "group memory" where nodes and links created during the collaborative process can be accumulated. Collaborative hypertext systems should support "collaborative intelligence where the group can obtain a far more intelligent result than the most intelligent member would have acting alone." [Turoff et al., 1991].

An application called InterNote has been implemented in the Intermedia framework to support annotative collaboration [Catlin et al., 1989]. The annotative collaboration process involves commenting, questioning and critiquing others’ work. In addition, these notes or annotations can be assimilated back into the original document to create a new and improved version of the document. The facility to incorporate annotations into the original document, resulting in data transfer across the annotation link, is called ***warm linking***. The system also supports annotation management, simultaneous multi-user annotation, and contention management.

The Virtual Notebook System (VNS) is a collaborative environment for scientific groups engaged in basic and clinical research in an academic medical center [Shipman III et al., 1989]. This system supports information acquisition, information linking and sharing, and information management in a group environment. It is capable of maintaining information about hypotheses, notes, and other relevant information which can be shared with other researchers located at remote sites. The system has facilities to import information from external sources such as electronic mail, USENET bulletin boards, bibliographic services, genetic sequencing information, and clinical databases.

Researchers who developed ABC, a hypermedia system for Artifact-Based Collaboration, consider the output of a group problem-solving process as an artifact [Smith & Smith, 1991]. For example, in the software development process, the artifact consists of concept papers, architecture, requirements, design specifications, programs, diagrams, references, and user manuals. The system can manage source code trees (as hypertext graphs), decomposition diagrams, call-flow diagrams, and uses-hierarchy diagrams. ABC contains six key components - a graph server, a set of graph browsers, a set of application programs, a shared window conferencing facility for synchronous and asynchronous communications, real-time audio and video, and a set of protocol tools to study group behavior and strategies.

6. Decision Support Systems and Issue Based Information Systems

Incorporating hypermedia with Decision Support Systems can greatly enhance the decision making process. This can be achieved by adding computation and dynamism to hypermedia. A knowledge-based Decision Support System called Max has been developed which incorporates hypermedia facilities to navigate among DSS application models, data, and reports [Bieber, 1993]. It supports browsing by analysts as well as executives. Analysts can execute decision models under various scenarios. Reports are dynamically created and they can incorporated with other models, data, and reports to produce summaries containing embedded generalized hypertext links. On the other hand, executives can browse through these final reports and follow hypertext links, if necessary, to look at information supporting an analyst's recommendations and findings [Bieber & Kimbrough, 1991].

Max contains the interface subsystem and the application manager subsystem. The interface subsystem contains the hypermedia engine and it is responsible for the dynamic creation and display of interactive hypermedia documents and menus based on application requests. The application manager subsystem provides the interface between the hypermedia engine and the underlying DSS applications such as finance, engineering, manufacturing, etc. These applications contain domain specific decision models (mathematical models) and data. This subsystem also provides the commands to execute the models, provide explanations, and generate reports. Elements in the application's knowledge base are mapped to components in the hypermedia engine through translation routines called bridge laws (See Chapter 6).

Issue Based Information Systems (IBIS) help members of a project team discuss issues related to a problem and come to a consensus on a solution. The graphical Issue Based Information System (gIBIS) from MCC is one such system which was built to capture the design rationale for software projects. Software design is a collaborative process in which various team members contribute their expertise and viewpoints to discuss the design. In gIBIS, participants in the online discussion argue about design issues by taking positions and making arguments for and against those positions. These position arguments are represented in a hypertext structure with three types of nodes: issues, positions, and arguments. Users can display overview diagrams to glance at the design rationale and look at the underlying text if required.

7. Software Engineering

Software design and development is accomplished through the cooperative efforts of team members who have to maintain an active information base in order to improve communication and coordination. Such an active information base can be achieved by applying hypertext to the Software Development Life Cycle (SDLC) [Balzer et al. 1989]. A number of documents are produced during different phases of the life cycle. These can be integrated together to form a highly cross-referenced body of information. For example, a design module in the design statement can be tied to its appropriate requirements specification and also to the relevant fragment of source code. Fragments of code can be linked to technical manuals. Similarly, several tools and concepts of software engineering can be applied to the development, use, and maintenance of hypertext systems.

DynamicDesign was developed at Tektronix as a CASE tool based on hypertext [Bigelow & Riley, 1987]. It is used to store and link C source code, requirements specifications, and other documents. A utility called graphBuild converts C source code into a hypertext graph based on the program’s call tree. A data dictionary is built for the program containing its local and global variables. It also contains sourceBrowse, a browser that allows the developer to traverse, view, and edit a source code tree.

The Documents Integration Facility (DIF) was developed at the System Factory, University of Southern California integrating a hypertext system with software engineering tools [Garg & Scacchi, 1987]. This "software hypertext system" was implemented to exploit the facilities of software engineering tools for automated software development and the unique storage and retrieval mechanisms of hypertext. Based on the experiences with DIF, it was realized that there was a need for developing a hypertext system that could utilize its knowledge about its users and their software tasks and products. Such a system would actively participate in the software development process rather than being just a passive storage facility. The researchers later developed the Intelligent Software Hypertext System (I-SHYS). The knowledge about the environment was partly embedded in the design of I-SHYS while the remaining was defined during the use of I-SHYS.

8. Other Applications

8.1 Simulation and Modeling

The hypertext paradigm has been applied to a biological research problem to study the energetics model of Cassin’s Sparrows [Schnase & Leggett, 1989]. This model tracked the daily energy expenditure of individual, adult sparrows. The computations and book-keeping activities associated with the simulation of the model and the analysis of field data were performed in an integrated hypertext environment using KMS on Sun workstations. Apart from hypertext related activities such as browsing, authoring, and annotating, researchers could write and run simulation programs. The resulting tables and figures could be integrated into the hypertext. This research proved that computational hypertext is appropriate for scientific applications, management of personal and group information, and community scholarship.

8.2 Law

Attorneys need a powerful information processing facility to gather information about cases, cross-reference them appropriately, retrieve them at great speed during trials and represent their ideas efficiently. They also need tools to make notes in a courtroom which can be shared with other attorneys to work on cases more effectively. In one law firm, it was found that all these activities could not be managed and linked even though there were a plethora of systems for word processing, electronic mail, billing, file management, and relational databases. Hypertext was used in this law firm to efficiently manage information about intellectual property. A system called HyperLex was developed using KMS to assimilate intellectual property information from various sources such as patents, trademarks, copyrights, trade secrets [Yoder & Wettach, 1989]. Information collected in this manner was properly organized, and cross-referenced to produce legal contracts, patent applications, court briefs and motions, and legal advice to clients. The system also provided links to group bulletin boards, calendars, employment agreements, and information on other previously published works. This system was used to catalog documents for a trial in which more than 10,000 documents had to be managed.

8.3 World Wide Web

World Wide Web (W3) is a hypertext-based information retrieval mechanism providing information access across heterogeneous platforms mainly connected over Internet [Berners-Lee, 1992]. It is based on the philosophy that information should be freely available to anyone. We can say that it is a small step towards Nelson's vision of a docuverse. The W3 architecture has allowed many existing hypertext systems and information bases to be incorporated as part of the web by gateway servers. W3 was developed for High Energy Physicists at CERN to share information with their counterparts across the rest of the academic community. It has now been extended to cover over 80 topics from Aeronautics to Social Sciences. W3 is based on a client/server architecture and the information (subject files) can reside anywhere on the network. Browsers use the File Transfer Protocol (FTP) and they can be installed at local sites to access remote webs. Each browser can handle, at a minimum, plain text as well as simple SGML formats.

8.4 User Interface Design

Hypertext can be used to prototype the user interface for any interactive system since horizontal prototyping (look and feel) involves designing windows (treated as nodes) and linking them together through menus and buttons and presenting the entire interface as a set of navigation paths [Nielsen, 1990].

8.5 Organizational Hypermedia

Organizational hypermedia is hypermedia technology applied to the information processing needs of an organization where information is shared among individuals [Isakowitz, 1993]. Daft and Lengel had stated that uncertainty and equivocality are two problems faced by present-day organizations. Hence, they acquire more information in order to reduce uncertainty. However, this does not solve the problem of equivocality or ambiguity inherent in organizational tasks. Minimization of equivocality requires face-to-face discussions and rich exchange of views among decision makers. Both uncertainty and equivocality can be reduced by employing hypermedia functionality in organizational information systems. Since hypermedia provides the structural mechanisms to manage complex relationships between various pieces of information, it can support equivocality reduction (similar to CMC systems). However, experiments are required to support these theories. Similar to group memory in a collaborative system, an organization manages information in a repository called "organizational memory". Organizational memories contain entities such as facts, positions, and events which are highly suitable for hypertext representation. However, hypermedia systems should address some of the problems associated with organizational memory such as pollution, growth, waste, restructuring, interoperability, and flexibility.

The following are some of the research issues related to organizational hypermedia:

The task perspective: Identification of applications based on tasks which are best suited for hypermedia implementation.

The knowledge perspective: Since most organizations handle knowledge/information, hypermedia should be able to represent and manage knowledge structures.

The integration perspective: Organizations have invested a great deal of resources towards many existing information systems. Introduction of hypermedia functionality should be seamless across these systems and hypermedia systems.

9. Summary

The hypermedia paradigm can be extended to many traditional information management systems as well as emerging complex information systems. Some of the application areas that can be greatly improved by incorporating hypermedia technology include online documentation, electronic encyclopedias, interactive kiosks, learning systems, idea processing environments, decision support systems, collaborative systems, issue based information systems, software engineering, and medical information systems.

References

[Balzer et al., 1989]. Balzer, Robert, Begeman, Michael, Garg Pankaj K., Schwartz, Mayer, and Shneiderman, Ben. Panel Discussion on Hypertext and Software Engineering, Proceedings of Hypertext ’89, ACM Press, 1989.

[Berners-Lee, 1992]. Berners-Lee, Tim. An Architecture for Wide Area Hypertext, Hypertext ’91 Poster Abstract, SIGLINK Newsletter, December 1992.

[Bieber, 1993]. Bieber, Michael. Automating Hypermedia for Decision Support, Hypermedia, Volume 4, Number 2, 1993.

[Bieber & Kimbrough, 1991]. Bieber, Michael and Kimbrough, Steven O. On Generalizing the Concept of Hypertext, Management Information Systems Quarterly, 1991.

[Bigelow & Riley, 1987]. Bigelow, James and Riley, Victor. Manipulating Source Code in DynamicDesign, Proceedings of Hypertext ’87, ACM Press, 1987.

[Catlin et al., 1989]. Catlin, Timothy, Bush, Paulette, Yankelovich, Nicole. InterNote: Extending a Hypermedia Framework to Support Annotative Collaboration, Proceedings of Hypertext ’89, ACM Press, 1989.

[Friedlander, 1988]. Friedlander, Larry. The Shakespeare Project Interactive Multimedia, Editors Sueann Ambron and Kristina Hooper, Microsoft Press, 1988.

[Frisse, 1988]. Frisse, Mark E. Searching For Information in a Hypertext Medical Handbook, Communications of the ACM, July 1988.

[Garg & Scacchi, 1987]. Garg, Pankaj K., and Scacchi, Walt. On Designing Intelligent Hypertext Systems for Information Management in Software Engineering, Proceedings of Hypertext ’87, ACM Press, 1987.

[Halasz, 1988]. Halasz, Frank G. NoteCards: A Multimedia Idea Processing Environment, Interactive

Multimedia, Editors Sueann Ambron and Kristina Hooper, Microsoft Press, 1988.

[Irish & Trigg, 1990]. Irish, Peggy M., and Trigg, Randall H. Supporting Collaboration in Hypermedia: Issues and Experiences, The Society of Text, Edited By Edward Barrett, MIT Press, 1990.

[Isakowitz, 1993]. Isakowitz, Tomas. Hypermedia, Information Systems, and Organizations: A Research Agenda, Proceedings of the Twenty-Sixth Annual Hawaii International Conference on System Sciences, January 1993.

[Maurer & Tomek, 1990]. Maurer, Hermann, and Tomek, Ivan. Broadening the Scope of Hypermedia Principles, Hypermedia, Volume 2, Number 3, 1990.

[Nielsen, 1990]. Nielsen, Jakob. Hypertext and Hypermedia, Academic Press, 1990.

[Raymond & Tompa, 1988]. Raymond, Darrell, R., and Tompa, Frank Wm. Hypertext and The Oxford English Dictionary, Communications of the ACM, July 1988.

[Shipman III et al., 1989]. Shipman, Frank M., Chaney, R. Jesse, and Gorry, G. Anthony. Distributed Hypertext for Collaborative Research: The Virtual Notebook System, Proceedings of Hypertext ’89, ACM Press, 1989.

[Schnase & Leggett, 1989]. Schnase, John L., and Leggett, John J. Computational Hypertext in Biological Modelling, Proceedings of Hypertext ’89, ACM Press, 1989.

[Shneiderman & Kearsley, 1989]. Shneiderman, Ben and Kearsley, Greg. Hypertext Hands On! An Introduction to a New Way of Organizing and Accessing Information, Addison Wesley Publishing Company, 1989.

[Smith & Smith, 1991]. Smith, John B., and Smith, F. Donelson. ABC: A Hypermedia System for Artifact-Based Collaboration, Proceedings of Hypertext ’91, ACM Press, 1991.

[Streitz et al., 1989]. Streitz, Norbert A., Hannemann, Jorg, and Thuring, Manfred. From Ideas and Arguments to Hyperdocuments: Travelling through Activity Spaces, Proceedings of Hypertext ’89, ACM Press, 1989.

[Turoff et al., 1991]. Turoff, Murray, Rao, Usha, and Hiltz, Starr Roxanne. Collaborative Hypertext in Computer Mediated Communications, Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences, January 1991.

[Yankelovich et al., 1988]. Yankelovich, Nicole, Smith Karen E., Garrett, L. Nancy and Meyrowitz, Norman. Issues in Designing a Hypermedia Document System, Interactive Multimedia, Editors Sueann Ambron and Kristina Hooper, Microsoft Press, 1988.

[Yoder & Wettach, 1989]. Yoder, Elise and Wettach, Thomas C. Using Hypertext in a Law Firm, Proceedings of Hypertext ’89, ACM Press, 1989.

CHAPTER 8

A SYSTEMATIC APPROACH TO USER INTERFACE DESIGN FOR A HYPERTEXT FRAMEWORK [*]

Abstract

A number of navigational tools exist for hypertext systems. Authoring guidelines have also been proposed for the organization of information in hypertext systems. However, there has been no systematic and comprehensive approach towards the design of user interfaces for hypertext systems. This paper is an attempt to apply a set of user interface design guidelines to a hypertext framework based on a cognitive model. This framework had classified nodes and links into various semantic types. We believe that such a classification is of great importance in developing an appropriate design metaphor/user interface for a hypertext system. A systematic approach to user interface design would also reduce functional opacity and system opacity.

Keywords: Hypertext, User Interface Design, Cognitive Models, Guidelines.

1. Introduction

The issue of incorporating semantics into a hypertext network has been addressed before [7,25]. Rao and Turoff had proposed a general framework for hypertext functionality based on Guilford’s Structure of the Intellect Model [20]. They had observed that as hypertext databases grow in size, they suffer from a lack of coherence due to ambiguity in meanings assigned to nodes and links. Hence, they proposed a framework which classified nodes into six different semantic types and links into twelve different types. Such a comprehensive framework would help designers develop better design metaphors and implementation models for hypertext systems. A first step in this direction is to develop an appropriate user interface which would reduce functional and system opacities.

A number of tools have been developed for hypertext navigation since Conklin addressed the issues of disorientation and cognitive overhead [8]. These include graphical browsers, overview diagrams, web views, paths, trails, guided tours and tabletops, history lists, timestamps, footprints, bookmarks, backtracks, queries, embedded menus, fisheye views and roam and zoom techniques [3,7,11,13,17,24,28,30]. Authoring guidelines have also been proposed for the organization of information in a hypertext system [23]. However, there has been no systematic and comprehensive approach towards the design of user interfaces for hypertext systems. This paper attempts to explore the application of a set of user interface design guidelines to the aforementioned hypertext framework.

2. Principles of User Interface Design

Designing modern user interfaces requires an in-depth understanding of various cognitive and mental models. One of the most widely used cognitive task analysis models in the area of user interface design is the GOMS (Goals, Operators, Methods, and Selection Rules) model [5]. It states that users employ certain established methods or procedures to achieve a pre-defined set of goals using operators and selection rules. The KLM (Keystroke-Level Model) attempts to predict time measures for error-free performance of tasks based on the sum of time taken for keystroking, pointing, homing, drawing, thinking, and waiting for the system to respond [21]. Both these models deal with error-free operation and do not address uncharted anomalies of operations when there are multiple goals and constraints [10].

The GOMS model and KLM models are suitable for tasks with well-defined structure such as text-editing or other linear tasks. However, they are not suitable for tasks with high cognitive variability such as navigating through a hypertext network. Hence, alternative models or approaches are required to develop user interfaces for hypertext systems. A first step in this direction is to develop a good user interface in order to reduce disorientation and cognitive overhead. This requires an understanding of the organizational setting, the targeted task domain, the typical user population, and the desired outcomes of hypertext navigation.

Various user interface design techniques and usability principles have been proposed by researchers and practitioners [1,15,18,19,21,26]. We have chosen the following set of guidelines [27] to be applied to the hypertext framework since they were found to be more systematic and comprehensive than other similar guidelines:

1. Identify the metaphor.

2. Identify all objects that make up the system.

3. Identify all actions/functions that can be performed on these objects. Separate them into generic actions, explicit actions, and control functions.

4. Identify modifiers/filters that select subsets of objects.

5. Identify strategic choices which allow the user to accomplish a specific task.

6. Identify lateral classifications - objects and actions that relate to each other.

7. Identify the formats of objects, parts of objects, menus etc.

8. Identify lists of objects.

9. Identify reactive choices that can be performed on a list or a set of objects.

10. Identify processes or functions that share information.

11. Identify all user interaction states.

12. Identify necessary help throughout the system.

13. Identify all error conditions.

14. Identify the screen layout - workspace, control area, status area, message area etc.

3. User Interface Design Guidelines for a Hypertext Framework

This section describes the user interface guidelines applied to the hypertext framework proposed by Rao and Turoff. The classification of nodes and links into various types makes it easier to design an appropriate user interface for a hypertext system.

3.1. Identify the metaphor

Carroll and Thomas have established that people develop new cognitive structures by metaphorically extending old ones [6]. Users of a new computer system can master it if they can metaphorically extend it to some real world objects or entities. A good metaphor not only helps the user, but also provides a rigid framework within which the hypertext author or designer must work to maintain consistency. Choosing an appropriate metaphor would also reduce both functional opacity (mismatch between the framework and the metaphor) and system opacity (mismatch between the metaphor and the implementation model) [20]. Hypertext has been compared to electronic encyclopedia, notecards, journeys, browsing, windows, paths, guided tours, travel holiday, and survey-type maps [9,12,14]. The travel metaphor serves as an extremely powerful aid to hypertext navigation [2]. At the same time, metaphors should not become too restrictive. Hammond and Allinson say that "the system should improve upon the metaphor, not be bounded by it." [12].

The metaphor suggested by this framework is "the general cognitive model of how individuals think about complex problems." [26]. In order to understand hypertext, designers must understand writing and reading models. According to the Cognitive Framework for Written Communication, writing is a combination of three activities: exploring, organizing, and encoding [22]. Writing is the transformation of a network of related concepts (retrieved from long-term memory or external sources) into an outline or a hierarchy which is later encoded into a linear sequence of words, sentences, paragraphs, sections, chapters, and illustrations. Reading is the execution of the above three processes in the reverse order. That is, a linear sequence of text is transformed into a hierarchy which is later integrated into a network in long-term memory. Thus, both reading and writing processes are based on the non-linear nature of thinking, a natural process in human beings. Human cognition is essentially organized as a semantic network in which concepts are linked together by associations. Hypertext systems should try to exploit this basic nature of cognition.

3.2. Identify all objects that constitute the system

In the hypertext framework based on the Structure of the Intellect model, both nodes and links can be considered as objects. Nodes are classified into six semantic types - detail, collection, proposition, summary, issue, and observation. Links are of two major types - Convergent and Divergent. Convergent links can be further classified into specification, membership, association, path, alternative, and inference links. Divergent links can be categorized into elaboration, opposition, speculation, branch, lateral, and extrapolation links. The emphasis of this framework is the association of semantics to nodes and links. Organizing nodes and links semantically helps manage the hypertext network and its sub-networks better. It would also help reduce ambiguity, disorientation, and cognitive overhead.

A collection node can be considered to be a composite node that can be comprised of details, propositions, summaries, issues, observations, and collections. A hypertext template can be considered as a collection node since it is defined as a set of pre-linked documents that can be duplicated [4]. The usage of a template will greatly speed up the process of an average user's understanding of the model or the metaphor. Templates automate the process of creating hypertext collections by creating skeletons of documents and linking them. Without a template, a hypertext author will have to start constructing the hypertext collection of ideas from the beginning. Many applications such as collaborative writing, collaborative hypertext, teaching aids etc., will greatly benefit from the concept of a collection node or hypertext template.

3.3. Identify all actions that can be performed on the objects

Actions and functions are the operations that can be performed on these 6 node types and 12 link types. Actions can be generic, explicit, or control-like.

3.3.1 Generic Actions

The ability to create webs of information from nodes and links between nodes is an inherent property of hypertext. Facilities to add, modify, and delete nodes and links can be considered as generic actions. That is, though these actions have a common meaning, their execution depends on the kind of object they act on. For example, deletion of a proposition node must delete all associative and speculative links to and from it. Similarly, deletion of an elaborative link must delete the footnote (a detail node) related to another detail node. Thus, the delete operation must behave differently based on whether the object is a node or a link. Similarly, a search operation can behave differently for different kinds of objects. Generic actions can also include the ability to create, edit, duplicate, or delete templates. Duplication of a template involves creating empty documents or nodes and the links or link sets associated with them.

3.3.2 Explicit Actions

Explicit actions are those which require an explicit qualifier or a modifier. The list operation is an explicit action since it requires the type of an object or other information. The result of listing all detail nodes can be a list in short form with object names or identifiers. A slightly informative version can be a preview of object names with all links to and from them and all nodes connected to them. A more complete explicit function can be the listing of nodes with their contents and all other nodes connected to them. The index operation can be provided by associative links in this framework. Explicit actions on a template can include the ability to add contents to empty documents, listing templates and their constituent documents and links, looking at an overview of the template, accessing a template by its type ("get a copy of the planning template") etc.

3.3.3 Control Functions

Control functions provide the ability to zoom into a subset of the hypertext network, forward and backward navigation through the web, provide overview maps, roam and zoom techniques, paths, trails etc. Control actions on a template can include displaying an overview of the template, zooming into specific link sets or webs (or "specific subgraphs") and looking at the contents of documents.

3.4. Identify all modifiers that select subsets of objects

The framework identifies all nodes and links by their semantic types. This type information can become a modifier or qualifier to the specific object. For example, a search can be performed on all nodes of type "detail". Similarly, we can also request for all links of type "specification". Type information and keyword information can be combined to narrow down the search criteria. For example, there can be a query to retrieve all nodes of type "detail" containing the keyword "hypertext". Modifiers for templates can include the type of a template, the author’s name, creation date etc.

3.5. Identify all strategic choices

Strategic choices include user interaction with the system in order to accomplish a specific task. These can be treated as landmarks [17]. Strategic choices might include overview requests (displays of summary nodes), structural and content query facilities, navigation mechanisms, editing tools, display options, audit trail mechanisms, linearization techniques, and backtracking facilities. The inherent nature of hypertext is that there is really no need for strategic choices. Any node or link in the entire network can be a strategic choice. In a true hypertext system each node should be self sufficient and complete. Also, the set of strategic choices need not be the same every time the user interacts with the system. For example, the user can directly access required nodes through query mechanisms. This feature must be available even without traversing the network. Hence, a query facility can be considered as a strategic choice. Strategic choices can also include an overview diagram of the template (its contents and links), the ability to find out from which master template a duplicate was created, the ability to edit the master template etc.

3.6. Identify lateral classifications

Hypertext provides the inherent capability of creating lateral classifications. The ability to create a lateral link to another version of a node or to an annotation (a detail or observation node) can be considered as lateral classification.

3.7. Identify the formats of objects and parts of objects

Though this section is more applicable only when designing the actual interface for a particular application, the designer must decide on the formats for nodes and links. Formats should also be defined for overview diagrams, indexing mechanisms, query facilities, and results of a query. A template can be considered as a pre-formatted collection of webs that can be directly used by the author.

3.8. Identify lists of objects

This is related to identifying explicit actions and modifiers. It is necessary to identify the kinds of lists that might be produced - lists of nodes or links or templates classified by semantic type, lists of nodes last visited, lists of nodes or links last modified etc. Lists can also include user created annotations.

3.9. Identify reactive choices that can be performed on a list or a set of objects

The ability to directly interact with (or manipulate) a node or a link in an overview diagram and view information contained within a node (or traverse to the next node) can be considered as a reactive choice. Previewing a node and the links to and from it (which can be highlighted showing a subset of the network) is another reactive choice. The ability to mark a node which has been visited using timestamp or footprint mechanisms is another reactive choice. Other reactive choices include marking a text item, creating elaboration links, and selecting from a list of objects. The ability to mark and annotate will help the user to store and manipulate retrieved information in order to integrate with other work. Reactive choices can include the ability to directly manipulate the contents of documents within a template such as editing, deleting, creating new links etc.

3.10. Identify shared processes

The ability to create objects (whether nodes or links) or list them is common across the system irrespective of the kind of object. Hence, these can be shared processes reacting differently based on the type of object being acted upon (similar to the object-oriented concept of operator overloading). The shared process of creation can be extended to create templates.

3.11. Identify all user interaction states

The designer must identify all possible interaction states, navigation paths, list requests, queries, backtracks etc. The user should always be informed about the current interaction state so as to minimize disorientation. The system should also allow the user to "peek" at a destination (before reaching it) by highlighting all the nodes and links connected to the specific node. The system can also inform the user about the node and link types in the user’s language. For example, if a detail node has an elaboration link to a footnote (another detail node), then the user should be informed through a feedback mechanism - "See associated footnote for further explanation of this concept". All user interactions can be stored in a history file so that the user can revisit the previously visited nodes the next time he or she brings up the system. Guided tours can be considered as a set of pre-defined interaction states or trails. Restoring previous interaction states to the way they were originally seen is very important for backtracking mechanisms [16].

3.12. Identify necessary help

Context-sensitive help is very essential in this kind of an environment to reduce disorientation and cognitive overhead. In this context, all the six node types and twelve link types must be well defined with examples so that users understand the exact classification of information. Help can be provided in terms of examples of hypertext collections created out of templates. Such templates can also be used for personal notes or annotations. Help can also be in the form of feedback on interaction states, reactive choices available at a particular state, navigational cues etc.

3.13. Identify all error conditions

All possible error conditions should be identified - it is better to prevent errors than "handle" them. There should not be menu choices or options that are not applicable in a particular interaction state. Error messages should be precise and constructive so that the user is informed of the exact cause of the problem and the steps to be taken to recover from it [15].

3.14. Identify the screen layout

Although screen design is application dependent, appropriate graphic design principles and usability principles must be used in designing the actual screen layout. The menubar should contain strategic choices as menu elements that are consistent across different components of the system and across different interaction states. The workspace or canvas should contain overview diagrams and displays of information contained within a node. The feedback area should be used to keep the user informed about interactions, status of queries, updates etc. The control area should provide facilities to zoom, pan etc.

4. Discussion

Rao and Turoff had claimed that all current hypertext systems fall under their hypertext framework and their semantic morphology can be extended to all future systems [20,26]. Since the user interface guidelines have been applied to such a general framework, we believe that these guidelines encompass user interface designs for all current and proposed hypertext systems. The next step is to determine the effectiveness of this approach by developing a user interface prototype for the discussed hypertext framework and conducting usability experiments.

5. Conclusion

Very little work has been done towards a systematic and comprehensive approach to the design of a user interface for a hypertext system using traditional and modern user interface design principles. This paper proposed the application of a set of user interface design guidelines to a hypertext framework developed earlier. It has been suggested that the understanding of how people use their cognitive skills to handle information will greatly enhance the design and evaluation of hypertext systems [29]. We believe that a user interface design based on the above systematic approach combined with the semantic richness of nodes and links would greatly reduce ambiguity, disorientation, and cognitive overhead. We need such an approach in order to integrate hypertext applications into our daily work in an increasingly collaborative environment.

References

1. Apple. Human Interface Guidelines: The Apple Desktop Interface, Addison-Wesley Publishing Company, Inc., 1987.

2. Baird, Patricia and Percival, Mark. Glasgow Online: Database Development using Apple's HyperCard, Chapter 5, Hypertext: Theory Into Practice, Edited by McAleese, Ray., Ablex Publishing Corporation, New Jersey, 1989.

3. Beard, David & Walker, John. Navigational Techniques to Improve the Display of Large Two-Dimensional Spaces, University of North Carolina at Chapel Hill Technical Report TR87-031, November 1987.

4. Catlin, Karen S., Garrett, Nancy L., and Launhardt, Julie A. Hypermedia Templates: An Author's Tool. Proceedings of Hypertext'91, ACM Press, 1991.

5. Card, S.K., Moran, T.P., and Newell, A. The Psychology of Human-Computer Interaction, Lawrence

Erlbaum Associates, Hillsdale, New Jersey, 1983.

6. Carroll, John M., and Thomas, John C. Metaphor and the Cognitive Representation of Computing Systems, IEEE Transactions on Systems, Man, and Cybernetics, March - April 1982.

7. Collier, George H. Thoth-II : Hypertext with Explicit Semantics, Proceedings of Hypertext ’87 Conference, November 1987.

8. Conklin, Jeff. Hypertext : An Introduction and Survey, IEEE Computer, September 1987.

9. Edwards, Deborah M. and Hardman, Lynda. "Lost In Hyperspace": Cognitive Mapping and Navigation in a Hypertext Environment, Chapter 7, Hypertext: Theory Into Practice, Edited by McAleese, Ray., Ablex Publishing Corporation, New Jersey, 1989.

10. Grant, Simon and Mayes, Terry. Cognitive Task Analysis ?, Chapter 6, Human-Computer Interaction and Complex Systems, Edited by Weir, George R.S., and Alty, James L., Academic Press, California, 1991.

11. Halasz, Frank. Reflections on NoteCards : Seven Issues for the Next Generation of Hypermedia Systems, Communications of the ACM, July 1988.

12. Hammond, Nick and Allinson, Lesley. The Travel Metaphor as Design Principle and Training Aid for Navigating Around Complex Systems, People and Computers III, 1987.

13. Koved, Larry & Shneiderman, Ben. Embedded Menus : Selecting Items in Context, Communications of the ACM, April 1986.

14. McAleese, Ray. Overview and Questions For Readers [on Hypertext], Chapter 1, Hypertext: Theory Into Practice, Edited by McAleese, Ray., Ablex Publishing Corporation, New Jersey, 1989.

15. Molich, Rolf, and Nielsen, Jakob. Improving a Human-Computer Dialogue, Communications of the ACM, March 1990.

16. Nielsen, Jakob. Hypertext/Hypermedia, Academic Press, 1990.

17. Nielsen, Jakob. The Art of Navigating through Hypertext, Communications of the ACM, March 1990.

18. Nielsen, Jakob. Traditional Dialogue Design Applied to Modern User Interfaces, CACM, October 1990.

19. Open Software Foundation. OSF/Motif Style Guide, Prentice Hall, 1991.

20. Rao, Usha & Turoff, Murray. Hypertext Functionality: A Theoretical Framework, International Journal of Human-Computer Interaction, 1990.

21. Shneiderman, Ben. Designing the User Interface, Addison-Wesley Publishing Company, Inc., 1987.

22. Smith, John B., Weiss, Stephen F., and Ferguson, Gordon J. A Hypertext Writing Environment and Its Cognitive Basis, Proceedings of Hypertext ’87, ACM Press, 1987.

23. Thuring, Manfred, Haake, Jorg M., and Hannemann, Jorg. What’s Eliza Doing in the Chinese room ? Incoherent Hyperdocuments and How to Avoid Them, Proceedings of Hypertext ’91, ACM Press, 1991.

24. Trigg, Randall. Guided Tours and Tabletops : Tools for Communicating in a Hypertext Environment, ACM Transactions on Office Information Systems, October 1988.

25. Trigg, Randall & Weiser, M. TEXTNET: A Network Based Approach to Text Handling, ACM Transactions on Office Information Systems, January 1986.

26. Turoff, Murray, Rao, Usha, and Hiltz, Starr Roxanne. Collaborative Hypertext in Computer Mediated Communications, Proceedings of the Twenty Fourth Annual Hawaii International Conference on System Sciences, January 1991.

27. Turoff, Murray. How to do it, Lecture Notes for Design of Interactive Systems (unpublished), NJIT, 1991.

28. Utting, Kenneth & Yankelovich, Nicole. Context and Orientation in Hypermedia Networks, ACM Transactions on Information Systems, January 1989.

29. Wright, Patricia. Cognitive Overheads and Prostheses: Some Issues in Evaluating Hypertexts, Proceedings of Hypertext ’91, ACM Press, 1991.

30. Zellweger, Polle T. Scripted Documents : A Hypermedia Path Mechanism, Proceedings of Hypertext ’89 Conference, November 1989.

---

[*] This chapter is based on a paper submitted to Workshop on Information Technologies and Systems ’93. An abstract of this chapter has also been accepted as a poster for Hypertext ’93.

[*Index*] [*Previous*] [*Postscript*]

CHAPTER 9

SUMMARY OF RESEARCH ISSUES

Hypertext systems can be considered to be database applications which provide a unique, non-sequential, and flexible method of accessing information through navigation and exploration. The essential features of hypertext are nodes which contain information and links which connect related nodes. It is the linking capability which makes hypertext more powerful than conventional information systems. There are a number of research issues related to the design, development, and application of hypertext systems. This chapter summarizes all these issues that were explored in detail in the previous chapters. Some approaches to addressing these issues have also been presented. The issues that were addressed during the recently held Hypertext '93 conference are also reported.

The following is the legend for this chapter:

- ● Items considered research issues.
- ● Possible solutions/approaches to address the research issues.

1. Implementation Issues

We have not fully understood the cognitive aspects involved in implementing hypertext systems.

- ● It is very essential to understand reading and writing models in order to understand hypertext. In addition, we need to understand the cognitive processes involved in argumentation and decision making. This needs revisiting some of the cognitive models such as Toulmin and Rittel argumentation schema and Guilford's Structure of Intellect Model.

- ● Also, existing metaphors such as electronic encyclopedia, notecards, journeys, browsing, windows, paths, guided tours, travel holidays, and survey-type maps are too restrictive and do not fully exploit the true potential of hypertext. The metaphor for hypertext should be based on "the general cognitive model of how individuals think about complex problems." Hypertext systems should try to exploit the basic nature of human cognition which is essentially organized as a semantic network of concepts linked together by associations. This general approach allows any individual to adopt the appropriate mental model by employing self-generated metaphors in the context of a specific application while the implementation itself can be based on a general semantic model [Balasubramanian & Turoff, 1993]. Such an approach provides navigation and analysis of the underlying database independent of the specific application and the different mental models of individuals.

Converting linear text to hypertext has been a classic problem while dealing with large information spaces such as encyclopedias, training manuals and dictionaries. Attempts have been made to convert these printed material both by manual and automatic means.

- ● Automatic conversion of text to hypertext based on structural features alone is not sufficient.

More work is required in the area of automatic link construction based on lexical and semantic analysis of text. The concepts of shallow apprentice (automatic linking based on lexical analysis) and clever apprentice (based on semantic analysis) must be explored further. The concept of automatic link generation based on the pattern of previously created user-links can be explored. Such an algorithm can use the user's Link Profile [Chang, 1993]. Some researchers have suggested guidelines for both manual and automatic conversion. These guidelines must be applied and evaluated.

Very little work has been done in exploring the use of hypertext templates in areas such as collaborative writing, teaching etc.

- Hypertext templates facilitate the design, organization, and presentation of a collection of knowledge in the form of hypertext. Templates can also be used to capture pre-defined or well-established relationships in organizational tasks. The specifics of the contents within the template can be filled in by the author.

Traditional writing is associated with implicit guidelines backed by thousands of years of linear writing experience. On the other hand, hypertext writing is new to authors and it is easy to produce an incomprehensible hypertext document.

- Researchers have suggested some general guidelines for authoring hypertext documents. These include dividing a hypertext document into three components: the content part, the organizational part, and the presentation part. Experiments must be conducted to assess the effectiveness of such authoring guidelines in developing large-scale hypertext systems.

Information systems such as Decision Support Systems and Expert Systems require a dynamic implementation of hypertext by incorporating virtual structures, computation, and filters.

- The use of bridge laws to generate dynamic structures must be explored further. Tools are also required for developers to define bridge laws easily. Dynamism can also be achieved by employing link-resolving algorithms which generate links to nodes based on user interaction [Tompa et al., 1993]. For example, the user can click on a word and the system can search for the occurrence of the same word or a synonymous word in other documents/nodes and traverse to that node. These can be considered non-authored link markers/links. Links thus become similar to database keys: not only can individual hypertext links be stored as explicit pointers but they can also be resolved "on the fly" through the execution of more complicated algorithms.

Very few attempts have been made to linearize hypertext documents for the purpose of printing.

- A first step towards linearization would be an algorithm to traverse all the nodes visited by the user and produce a linear document. Another method would be for the user to specify the start and end points of linearization so that all interactions between these two points will be linear.

2. Database Issues

Database requirements of hypermedia systems have received very little attention. Relational Data Base Management Systems (RDBMS) cannot fully address the data handling requirements of hypermedia systems.

- Object-Oriented Data Base Management Systems (OODBMS) have been shown to meet the storage requirements of complex systems such as geographical information systems, multimedia systems etc. We need to investigate the use of OODBMS as storage mechanisms for hypermedia applications.

- Object-oriented approaches to hypermedia must be investigated. Many features from the areas of object-oriented analysis, design and programming can be extended to the hypertext model. Integrating an object-oriented data model with hypertext can greatly improve the information retrieval process. This can be accomplished by redirecting hypertext functionality from the application level to the database level using object-oriented database systems.

3. User Interface Issues

Although many designs exist for hypertext navigation, the problems of disorientation and cognitive overhead still persist. In a true hypertext system, users must be able to move freely through the system according to their needs, without getting lost either spatially or cognitively.

- The facilities to navigate through a hypertext database must be at least as rich as those available in books. Many user interface solutions have been developed by different groups of researchers. However, all these designs have been ad hoc approaches to navigation problems. There has been no systematic and comprehensive approach to user interface design for hypertext systems. A set of fourteen user interface design guidelines were presented in this review paper. In addition to following these guidelines, it will be highly beneficial to integrate existing navigational tools and study the effectiveness of such a systematic and comprehensive approach.

- Also, formal methods need to be developed for usability testing and evaluation of hypertext systems. Navigation techniques and evaluation measures must be based on the organizational setting, the targeted task domain, the typical user population, and the desired outcomes of navigation. Experiments must also be conducted to evaluate the effect of deliberately incorporating disorientation and cognitive overhead in learning systems in order to encourage exploration and learning.

4. Information Retrieval Issues

While navigation or browsing is sufficient for small hypertext systems, more powerful information retrieval and indexing techniques are required for large scale hypertext databases.

- A browsing session can take a long time before converging to the required item or may not converge at all. Therefore, browsing mechanisms must be supplemented with querying techniques. In addition to content queries which retrieve the contents of nodes there should be structural queries to retrieve subgraphs of the hypertext network that match a given pattern. Query facilities which incorporate both content search and structure search can act as filters. Also, the implicit structure present in documents in terms of spatial characteristics such as geometry, distance, collocation, recurrence etc., can be analyzed to retrieve templates and document outlines

[Marshall & Shipman III, 1993]. Media-based navigation and picture-index techniques must be explored further to retrieve objects based on shape, color, motion, and auditory features [Hirata et al., 1993].

- Many researchers have investigated the possibilities of separating index information from contents thus forming an index space (or concept network) on top of a content space (or document network). These would not only facilitate IR but also accommodate dynamic linking and independent maintenance of the two networks. Further research is required in the area of aggregating hypertext networks into semantic or hierarchical clusters.

Existing query languages are suitable only for content-based searches.

- Query languages need to be extended to perform structural queries. These extensions include the notions of quantifiers, recursive operators, aggregation, and improved semantics. Visual query languages need to be explored further. Such visual queries should balance expressiveness with ease of use. Users should be able to retrieve information by specifying spatial properties, templates, shapes, color etc. They should also be able to express queries by selecting and manipulating visual representations of hypermedia objects. Such a query language is being developed as part of the Multimedia Object Retrieval Environment [Lucarella et al., 1993]. It will be interesting to see the results of their work. More work is required in the use of belief networks or Bayesian inference networks for hypertext-based IR. The computational complexity of these approaches need further investigation.

Very little work has been done in the area of merging Artificial Intelligence with hypertext.

- A combination of inference-based IR and knowledge-based hypertext could greatly facilitate browsing and searching.

Formal methods and experiments are required to measure the effectiveness of these IR techniques.

5. Integration Issues

In order to make hypertext systems fully open and integrated, the following issues must be addressed: interoperability, programmability, node and link typing, distributed linking, concurrency control for multi-user access in a shared environment, maintaining public and private links, operating systems support, networking, bridge laws, linking protocols, multimedia support, user interface consistency, and version control. Most of these requirements can be addressed using object-oriented techniques.

- Interoperability can be achieved by employing some of the many layered, platform-independent architectures, models or engines, and frameworks that have been proposed and developed by researchers in an effort to make hypertext systems more generic and integrated into the desktop environment. In order to make hypertext systems fully portable, existing document standards such as ODA and SGML must be extended to support unstructured documents and linking. International standards such as HyTime and MHEG are emerging to support hypertext functionality and multimedia information in applications. The applications of these standards to operational and proposed systems must be investigated. A few commercial products such as FrameBuilder from Frame Corporation, DynaText from Electronic Book Technologies, and

PassageWays from Passage Systems provide SGML-based hypertext authoring tools [ComSymHT93, 1993].

- Programmability can be achieved by providing application development toolkits for adding hypertext functionality to existing systems.

- The concept of concurrency control is quite different in a multi-user collaborative environment as opposed to a multi-user environment. Such environments require complex concurrency control mechanisms such as event notification, fine-grained notification, shared locking, fine-grained locking, user-controlled locking, and persistent collaboration information [Wiil & Leggett, 1993]. The requirements suggested by these researchers must be applied to collaborative hypertext systems.

- Linking protocols such as Sun's Link Service, Intermedia's Link Server, Microcosm Link Engine have been developed. Some of these are closed systems. They work with only certain applications on certain hardware platforms/operating systems. The commercial use of these protocols must be explored further.

- Most linking protocols exist as layers above the operating system. Making the link service an integral part of the operating system must be investigated. Such an attempt is being made as part of the Macintosh implementation of the Microcosm Link Engine (called Macrocosm) [Lewis, 1993].

- Only when hypertext functionality becomes an integral part of our computing environment will knowledge workers accept and incorporate hypertext into their daily work process.

6. Applications

Most corporations seem to be interested in the profit-making and entertainment aspects of multimedia rather than intellectual applications of hypermedia such as learning systems, group decision support systems, brainstorming systems etc.

- The hypermedia paradigm can be extended to many traditional information management systems as well as emerging complex information systems. Some of the application areas that can be greatly improved by incorporating hypermedia technology include on-line documentation, electronic encyclopedias, interactive kiosks, learning systems, idea processing environments, decision support systems, collaborative systems, issue based information systems, software engineering, and medical information systems.

- Many corporations suffer from the lack of a comprehensive view of their data/information due to the proliferation of many database applications. These applications were developed over a long period of time, each addressing a specific problem. A hypermedia link engine would be a good mechanism to integrate such heterogeneous database applications providing unique navigation and information retrieval facilities to retrieve large amounts of inter-related information.

References

[Balasubramanian & Turoff, 1993]. Balasubramanian, V., & Turoff, Murray. User Interface Design Guidelines for a Hypertext Framework, Poster Abstracts, Hypertext ’93, 1993.

[Chang, 1993]. Chang, Daniel T. HieNet: A User-Centered Approach for Automatic Link Generation, Hypertext ’93 Proceedings, ACM Press, 1993.

[ComSymHT93, 1993]. Commercial Symposium, Hypertext ’93 Proceedings, ACM Press, 1993.

[Hirata et al., 1993]. Hirata, Kyoji, Hara, Yoshinori, Shibata, Naoki, and Hirabayashi, Fusako. Media-based Navigation for Hypermedia Systems, Hypertext ’93 Proceedings, ACM Press, 1993.

[Lewis, 1993]. Lewis, Andrew J. A Hypermedia Link Service as an Operating Systems Extension, Poster Abstracts, Hypertext ’93 Proceedings, 1993.

[Lucarella et al., 1993]. Lucarella, Dario, Parisotto, Stefano, and Zanzi, Antonella. MORE: Multimedia Object Retrieval Environment, Hypertext ’93 Proceedings, ACM Press, 1993.

[Marshall & Shipman III, 1993]. Marshall, Catherine C., & Shipman III, Frank M. Searching for the Missing Link: Discovering Implicit Structure in Spatial Hypertext, Hypertext ’93 Proceedings, ACM Press, 1993.

[Tompa et al., 1993]. Tompa, Frank Wm., Blake, G. Elizabeth, and Raymond, Darrell R. Hypertext by Link-Resolving Components, Hypertext ’93 Proceedings, ACM Press, 1993.

[Wiil & Leggett, 1993]. Wiil, Uffe Kock, Leggett, John J. Concurrency Control in Collaborative Hypertext Systems, Hypertext ’93 Proceedings, ACM Press, 1993.

[*Index*] [*Previous*] [*Postscript*]