KATHOLIC UNIVERSITY OF NIJMEGEN
FACULTY OF MATHEMATICS AND INFORMATICS

TECHNICAL UNIVERSITY OF PATRAS
COMPUTER SCIENSE DEPARTMENT

MASTER'S THESIS

ON ELECTRONIC PUBLISHING:
Exploring a new medium

A.VAGELATOS — A.HATZIMANIKATIS

Supervisor professors:

C.H.A.KOSTER — D.CHRISTODOULAKIS

OCTOBER 1989

Contents

1 Introduction

Imagine, walking into a big library and picking up a book on *Socrates.* You start reading and learn that Socrates was an ancient Greek philosopher. You wonder what else was happening in Greece then, so you go to the card catalog, find a book on Greek history, go to the stacks, locate the volume (if you are lucky and it's not checked out) and read it, before you continue.

In this book you find a reference to Athens and you wonder what Athens looks like now. Back to the card catalog, and the stacks, to find a book with images from Athens.

This process continues until you have either satisfied your desire for knowledge on the subject or the librarian kindly announces that it's too late and the library has to close.

Now imagine yourself sitting in a comfortable chair in front of your computer. You begin to read about Socrates. When you wonder about Greek history, you simply highlight the text and request information with a mouse click! To find images from Athens you use the same process.

The above is a rather idealized example of using computer power, to augment the human's access to world's literature.

Developments in computer technology now make it possible to store stupendous amounts of written material economically and efficiently. Many systems that manipulate (edit, store and retrieve) text are already either in use or in development. The "word processing" systems have already greatly changed the handling of the written word in the office environment. Now the question is: *How to extend this approach?*

This is our master's thesis, made at Katholic University of Nimegen (KUN), in cooperation with the Technical University of Patras. In this thesis we examine "what's going on" in this area, the area of *Electronic Publishing,* especially towards the use of computer power not only for preparing documents, but also for storing and presenting documents: *The step from printed media to electronic media.*
We also describe an Electronic Publishing system prototype we built.

In **Part I**, we examine Electronic publishing and Hypertext generally. Chapter 2 is about Publishing generally. Apart from that some definitions are also given. In chapter 3, we investigate Electronic publishing, its advantages and disadvantages over print medium.

Chapter 4 is about Hypertext and chapter 5 is an overview on most of the existing Electronic Publishing and hypertext systems.
In chapter 6 there is a classification of such systems and general comments about them.
Finally in chapter 7 there is a description of the *Online Compact Journal*, a prototype implemented in KUN in 1982, on which we based the implementation of our prototype.

In **Part II**, we describe the structure of our prototype, its features and finally we give some thoughts for the future.
Chapter 8 is a general description of our prototype.
In chapter 9 we discuss the main features of the system, while in Chapter 10 we give our thoughts about the user interface.
Chapter 11 is about the future orientations of Electronic Publishing systems.
Finally chapter 12 is the epilogue of our thesis.

At this point we would like to thank some people that have helped us with this thesis.

First C.H.Koster, professor of the *Faculty of Mathematics and Informatics* at Katholic University of Nijmegen, who was the supervisor of our work.

D. Christodoulakis, professor of the *Computer Engineering Department* at the Technical University of Patras, who in cooperation with C.H.Koster arranged the exchange under Erasmus program.

To all the members of the Faculty of Mathematics and Informatics at KUN, thanks for the friendly environment we found and worked in.

Particular mention must be made of Mathias Moritz and Joop Leo of the KUN for their constructive comments and criticisms on the draft of this thesis.

Part I

Electronic Publishing and Hypertext systems

2 Publishing

Publishing is an old and conservative field. Hundreds of years of development have gone into creating our methods of presenting written information, and all literate human being spend much of their childhood learning to read printed forms. Books maybe the greatest of human inventions.
But the world changes, and the means of producing printed material has changed completely several times in this century.

The latest technique used in the publishing world is the **Digital Typography** or **Electronic Publishing**. Digital Typography as [Rubinstein 88] defined it, is:

> The technology of using computers for the design, preparation, and presentation of documents, in which the graphical elements are organized, positioned, and themselves created under digital control.

But so far in these techniques the main focus has been on document preparation as a whole. The other half of the story, which has too often been ignored, concerns effective *reading* of a document. Reading is a rather direct process, but when references are involved a reader relies on a somewhat indirect approach. For instance, when a bibliographic reference is of interest, the reader needs to go to the bibliography and look up the cross-reference information available there.

The notion of documents as static still dominates our way of reading even in the area of electronic media. On our favorite document-preparation systems, we are still creating bibliographies and indexes in the traditional way. Part of the reason for doing so is the need for hard copies consistent with the traditional form. But if this requirement can be relaxed, we should think seriously about what exactly the purposes of references like a bibliography and index are. Their foremost function is to allow the reader to access relevant information efficiently. Creating separate bibliography and index sections is the best we can do in the static print medium.

In an integrated electronic document environment, much of this information can be stored internally. Instead of requiring the reader to actually read the section that contains the references (indirectly access the information), the system can allow the reader to access references in a *direct and context-sensitive way.* For instance, when a citation is of interest, the reader can use a menu of options to either inspect the content of the bibliographic entry in a separate window so that reading of the main document is not hindered, or visit the actual document referenced by the citation. If the reader selects an object of a different nature, its context gets reflected in the menu immediately.

Systems that support this type of nonlinear reading are referred to as *hypertext* systems. The key here is *the ability to create links among objects* within the same or different documents.

2.0.1 Term definition

It is now time to give the definitions of some terms that we use in this thesis.

We decided to use the term *Digital Typography*, as [Rubinstein 88] defined it (see page 7): using computer technology to produce traditional paper publications.

Unfortunately, the term *Electronic Publishing* is used with many different interpretations (i.e. in the publishing world, many people use this term with the meaning that we just gave to digital typography). Already since 1980 [Nelson 80] has used this term to denote: "public-access document systems with digital storage and demand services to the open public."

Documents have a lifecycle. Till now tools have been developed mainly for document preparation, using computer systems. This is what we call digital typography.

> By Electronic Publishing we mean using computer power to support all the phases of documents life cycle: preparation, presentation and maintenance (fig. 1).

Hypertext systems (based on the idea of non–linear writting and reading), fall in the category of Electronic Publishing systems. In the next chapter we will discuss in a greater detail Hypertext systems.

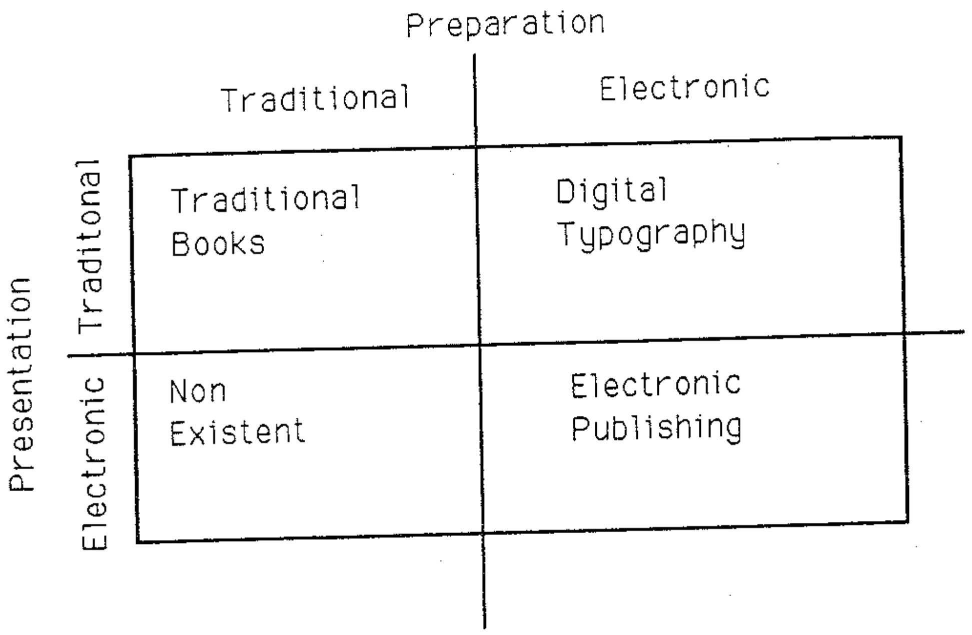

Figure 1: Different forms of publishing methods, using different preperation and presentation techniques.

3 The new form of Electronic Publishing

Examining the differences between the *print* and *electronic* media (hardcopy books and electronic books) is an interesting investigation that creates a useful framework for critically thinking about the computer's role in document preparation and presentation. Although electronic documents are not directly comparable to paper books, they do serve a number of common purposes. Both are used as sources of information, as learning devices, and as mechanisms for communication between people who are distant in time or place.

By considering the strength and weaknesses of paper and electronic document systems, it is possible to formulate a set of capabilities that electronic documents should possess to maximize the advantages of the electronic medium and overcome some of the disadvantages inherent in the print medium.

3.1 Print medium vs. Electronic medium

Scholars, or "knowledge workers", rely heavily on print media, even though electronic creation and dissemination of information is possible with today's technology. In some cases, this reliance on print is part of a long, ingrained tradition, but in other cases, print is still simply the most appropriate vehicle, either because electronic documents systems are still impractical to use or because they do not meet a particular objective as well as does paper.

The most important fundamental property of books is that they are *static*. Once printed, a book cannot be altered, except by reprinting, and at no time do the readers have the opportunity to change or manipulate its contents. The static nature of books is both their biggest asset and their most serious shortcoming.

Electronic document systems have their own advantages and limitations. In some cases they are more powerful or appropriate than paper books i.e. for meeting the range of information needs of scholars within the university community. In other cases, books are more useful (imagine reading a novel story on your terminal!).

3.1.1 Advantages of Electronic Document systems

Theodor Nelson and Douglas Engelbart [Conklin 87] were among the first to articulate the benefits of electronic document systems. In the early 1960's, they recognized that computers were well suited to help scholars and others to create connectivity–webs of related information. In the printed medium scholars often mark up books, articles, and papers. When a phrase or illustration sparks a connection to an idea in another book in a scholar's mind, he or she writes that connection, or "link", down next to the phrase or picture that sparked the thought. Providing footnotes, references, and word glosses in books is an author's way of making annotations or explicitly indicating connections between his or her writing and other documents, schools of thought, and definitions. Many researchers (G. Landow, T. Nelson [Yankelovich 85]) see connections and the act of following links, as crucial to education. Electronic document systems help scholars both create *connections* and follow those made by others. Because they allow *flexible organization* of material, they provide authors and readers with a greater degree of freedom than printed books.

Linking scholars together –intercommunicability– is an essential aspect of *connectivity*. Electronic document systems running either on a multiuser, time-shared system or on a series of networked workstations allow authors and readers to *communicate* with one another in a number of ways. Colleagues can easily view one another's documents (if given permission), send and receive personal *electronic messages*, and jointly edit the same document without leaving their own workplace.

Another great advantage of electronic documents over paper ones, is their ability to handle many more *graphic elements*. By combining a variety of media, they can provide not only static images, but also dynamics (e.g., computer animations and computer–controlled video sequences), *interactivity* (e.g., ability to move objects, change and edit objects, and change states), and *sound* (e.g., computer–generated or audio disk recording).

The paper medium does not allow a reader to alter the contents of a book. Electronic documents, however, are *dynamic* in the sense that both authors and readers can customize the material contained within a corpus of documents. For example, in a military setting, an author may want to provide complete access to certain information to those with the appropriate security clearance and only partial access to all others. If working with paper, this author would be forced to publish separate books for each constituency.

Readers also may want to *filter* the information. For instance, a lawyer might want to apply a filter that would display all cases mentioning the name "J.Smith", or to view the first sentence of every major decision in a given area.

Finally, the electronic medium can aid dramatically in the *updating* and *distribution* of information. In many cases editing an electronic document is far more efficient than making changes to a printed book. In addition, the cost of distribution (in terms of both money and natural resources) one day may be greatly reduced by the advent of national networks and high-density storage devices (e.g., diskettes, video disks, CDs, etc.). And everyone can easily see that this day is not very far away.

3.1.2 Disadvantages of electronic document systems

There are two classes of problems with Electronic Document systems: problems with the current implementations and problems that seem to be inherent to such systems.

The problems in the first class arise from the limitations of hardware and software design, and include *delays* in the display of reference information, *restriction* on names or kinds of information that somebody can manipulate, etc. A major problem comes from the totally different way of presenting information in electronic document systems: The information comes through small (in most cases) computer screens with *limited resolution.* Another major shortcoming (in this class of problems) in most electronic document systems developed using current technology, is their failure to provide adequate information about where readers are in the document. Readers of paper books can always tell if they are "at the end of the book" or "three-quarters through it". In contrast, electronic document systems offer more degrees of freedom, more dimensions in which one can move, and hence a greater potential for the user to *become lost* or *disoriented.*
Aside from that there are some other disadvantages that arise from todays hardware's limitations. Many people complain about *eye-strain* from working at a computer, even with high-resolution graphic screens. *Cost* is still a major limiting factor to the widespread use of electronic document systems. Another problem arises because of the new evolutionary technology, this time. Traditional publishers limit the size of books for economic reasons and in accordance with market forces; with the power and the capacity of modern computer systems, there is a temptation to continue adding information and

links for no better reason than the fact that the space is available. This fact may lead to a document with a tremendous number of links and unuseful information which most of the times confuses the user.

The major problem in the second class, is that it is difficult to become accustomed to the additional mental overhead required to create, name, and keep track of links. This problem has two sides: The first related with the creation of the documents, and the second related with the reading of it.
Suppose you are writing about X, and a related thought about Y comes to mind and seems important enough to capture. Ideally an electronic document system allows you to simply "press a button" and make the connection. Unfortunately, the situation is a bit more complex Athan that. You have to consider about the naming of the link, which should be descriptive for the reader. Beyond that, you must also consider if there are better ways to link Y to the network of thoughts than at the point X, end a lot of problems like this.
The problem also occurs in the process of reading an electronic document. Such systems tend to present the reader with a large number of choices about which links to follow and which to leave alone. These choices engender a certain overhead of decision making for the reader. The problem is that, even if the system response time is instantaneous (which it rarely is), you have to pause for some seconds to consider whether to follow this side path or another.

In the following table (next page) we try to list the main advantages and disadvantages of Electronic documents (in comparison with print medium).

Electronic document system	Print medium
+dynamic in the sense that it is easy to change	* static (The static nature of books has both advantages (historical value, nobody can change the "historical facts" that have been written in a book) and disadvantages (a small change leads to publish again the particular book).
+gives the ability to the user to create connections between different articles (annotations)	– very limited, spoils the book
+more powerful in the sense that the information are always available (in contrast to a lend out book in a library)	– less powerful in that sense
+ability to handle many more data types, graphic elements, dynamic images, sound and even simulation	– can handle only text and static two-dimension images
+filtering: one can customize the material by providing different access authorizations	– no such ability
+easy and cheap updating of information	– here updating means reproduction
–fails (in the existing systems) to provide adequate information about where a reader is in a document and in the whole system (disorientation)	+ the reader can easily find his position in the document
–limitations of todays hardware	?
–additional mental overhead to create, name and keep truck of links	?
–	+ more "reliable hardware"
?	+ historical value
–	+ portable in the sense that you can carry a book everywhere
–People are not used to such systems	+ easy to read
–The only standard is that there are no standards	+ well defined and accepted standards

3.2 A taxonomy of documents

As we have already said, not all the types of documents can be adequately represented via electronic document systems. The reasons for that are:

- Documents have to be split up in small modular text entities which will be linked together.
- Documents are better suited to the electronic representation if they have a hierarchical organization.

What is required is an understanding of the types of documents that are appropriate for use with electronic document systems. We present some thoughts about such a taxonomy below.

The range of types of paper documents currently available is quite considerable: newspapers, magazines, novels, journals, reference books, encyclopedias and manuals, are the most important.

Electronic document systems can support a nonlinear structure of text, but is clear that for certain types of documents such nonlinearity is neither necessary nor desirable. For example, the detective novel is a literary form which relies heavily on its linear format; a philosophical argument frequently builds from earlier arguments in such a way that it would not be appropriate to jump into the middle without having encountered the earlier material.

On the other head a computer-science book is frequently arranged in chapters on different sub-areas, and a reader most of the times just wants to read a certain chapter without having to read the whole book.

Manuals seem to fall easily into the range of document types that suit to Electronic Document systems. A reason for that is their hierarchical organization. Apart from that information can be split into small, self contained entries which may have several cross-references to other entries. The top level can contain the various sections, each section being replaced by more detailed text and graphics. This way the user can access at first a short description of e.g. a command that he (she) would like and after that may proceed to a more detailed one or leave the system. Such a system was implemented by Symbolics Software [Walker 88] and is called *Document Examiner* (see also p. 25). It is an online manual for the Symbolics Workstations.

The *encyclopedia* is another document type that is well suited for electronic representation. The essence of an encyclopedia is that it is a collection of text units that are designed to stand alone as separate modules but which are enchanced by cross–references to other entries. This leads to a straightforward representation of encyclopedias in electronic document systems. Such a system is described in [Glushko 89]. It is a hypertext version of a multi-volume engineering encyclopedia on a compact disk. The text of this system occupies about 20Mb, with the graphics requiring another 150Mb. In addition, the disk contains another 40Mb for various indexes.

Electronic document systems seem to be also appropriate for University's communities, especially for *paper authoring*, *journal authoring*[1] and collaborative work. Many such systems (e.g. Neptune [Delisle 86], Notecards [Halasz 88]) have been developed with their main task the idea processing, the document authoring, and the software development. In such systems, tools for creation and modification of the text are well-developed. Versioning is also suplied.

Finally *newspapers* and *magazines* present far more information at a glance than do high-resolution displays. So, in the near future it seems rather impossible to see such kind of documents presented via an Electronic Document system.

[1]The main differences between journal, magazines and newspapers are the *frequency* of publishing (journal: low, magazine: medium, high: newspapers) and the *seriousness* (magazines are read for pleasure, journals for enlightment and newspapers for both).

4 What About Hypertext?

The idea of *Hypertext* or is not a new one. Theodor Nelson (who coined the term hypertext) and Douglas Engelbart were among the first who really realized the advantages of *hypertext* in the early 1960's. Then they realized that computers were well suited to help scholars and others create connectivity webs of related information. (Nelson became interested in hypertext because of his habit to take notes in every paper he was reading. And after that he wanted to be able to see the original context of the note.)

There is no general accepted definition of what Hypertext is. Conklin [Conklin 87] has written about that:

> The concept of Hypertext is quite simple: Windows on the screen are associated with objects in a database, and links are provided between these objects, both graphically (as labeled tokens) and in the database (as pointers).

These objects are in most of the cases units of text, generally called *nodes* but having many other names in the existing systems: *notecards, frames, notes, chunks, compacts,* ... The nodes may also contain other forms of information such as images or sound; then the term *Hypermedia* is used.

The essential feature of Hypertext systems are the *machine-supported links* [Conklin 87]. An additional feature that is common to many Hypertext systems is the heavy use of windows that have a one-to-one correspondence with the nodes.

The user accesses the information in the nodes by navigating through the links. Content searching should also be supplied, although many existing systems do not support this feature. By creating, editing and linking units, users can build *information structures* for various purposes (authoring documents, developing on-line help systems ...)

Hypertext, at its most basic level is a DBMS that lets you connect screens of information using associative links. At its most sophisticated level, Hypertext is a software environment for collaborative work, communication, and knowledge acquisition. Hypertext products mimic the brain's ability to store and retrieve information by referential links for quick and intuitive access.

Hypertext programs have been adapted for *Electronic Publishing, project management, system analysis, software development.*

The Data Base Management Systems (DBMS), relational, network or hierarchical was a revolution in the way information is organized in organizations and the way we think about it. Hypertext and Electronic Publishing systems seem to be a new revolution in the way textual information can be organized. They may change the way documents are created, read and distributed. They may even change the way we think and learn.

5 Existing Systems

In this section we give a brief overview about existing hypertext systems and we also mention some older systems having historical value.

5.1 First hypertext system

The first (non-computer-based) hypertext system was envisioned by Vannevar Bush in 1945 who pictured a "memex" device in which individuals would store all their books, records and communications, form trails through them, and rapidly retrieve specific informations contained within them.

5.2 The Brown University Research

5.2.1 FRESS

The File Retrieval and Editing System [Yankelovich 88] was a text–only system, allowing authors to create links within any text document, and was developed under VM/CMS time sharing system in the late 1960's. The system contained important navigational and linking facilities. It was designed for multiple users, although it didn't support concurrent updating of a single file. It also provided filtering facilities.
It had 2 major drawbacks:

- Commands to establish links were rather complex.
- No orientation facilities.

5.2.2 Electronic Document System (EDS)

The successor of FRESS, the Electronic Document System [Yankelovich 88] completed in 1982, represents Brown's (university's) first *Hypermedia* system. The system was developed on a DEC VAX 11/780 and uses a Ramtek 9400 color display screen plus a data tablet and a pointing device.
Its main improvements over FRESS are the high-resolution color graphic capability, its graphic representation of the information web, and its ability to incorporate animation sequences. It also has a filtering capability.
The most interesting aspect of the EDS is the ability provided for authors to represent the underlying information web, and two forms of maps provided for readers.

5.2.3 Intermedia

Intermedia [Yankelovich 88], currently(?) under development at Brown's Institute for Research in Information and Scholarship, is a *multimedia* system. It is a framework for a collection of tools that allow authors to make links between standard types of documents created with heterogeneous applications.
The system which runs on a network of Unix-based workstations currently contains five integrated applications: 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 applications conform to Macintosh/Microsoft Windows interface standards.
Intermedia allows the user to create bidirectional links from a specific location in one document to a specific location in another. There is also a search feature.

Intermedia differs from most other hypermedia systems in that it allows multiple users to both follow and create links concurrently in the same web (it has three access rights: read, write, annotation).
In addition to the linking capability a user can create paths (named trails) through a particular set of links. A special system path keeps track of the user's entire Intermedia sessions. The Intermedia user environment is still at a prototype stage.

5.3 Knowledge Management System (KMS)

KMS [Akscyn 88] is a distributed hypermedia system, based on the ZOG system which was developed earlier in a research project at Carnegie-Mellon University. Work on ZOG began at CMU in 1972. In 1980 ZOG was used on a major application project to build a computer-assisted management system for the U.S. Navy's newest nuclear powered aircraft carrier, the USS Carl Vinson. In March 1983 Carl Vinson left on its first deployment with a distributed ZOG system running on a network of 28 PERQ workstations.

That version of ZOG supported 4 applications:

- On-line policy manual (ship's organization and regulation manual)
- Interactive task management system (for analyzing and tracking complex tasks)

- On-line maintenance manual with interface to videodisk! (for weapon elevators)

- Interface to the AirPlan expert system

In 1980 (at the request of Westinghouse) the ZOG researchers formed a company (Knowledge Systems) to develop a commercial version of ZOG. This initial work led to the first version called KMS in 1983. KMS supports *organization-wide collaboration* for a broad range of applications, such as electronic publishing, on-line documentation, project management, software engineering and computer aided instruction.
The heart of KMS is its conceptual data model. A KMS data base (which is hierarchical) consists of screen-size workspaces called *frames* which contain text, graphics and image items. Individual items can be linked to other frames or used to invoke programs. The database can be distributed across an indefinite number of file servers and its size is only limited by the available disk space.
The KMS user interface employs a form of direct manipulation designed to exploit a three-button mouse. A combined browser/editor is used to traverse the database and manipulate its contents. Over 90 percent of the user's command interaction is direct; a single point-and-click designates both object and operation.
Running on Sun and Apollo workstations, KMS accesses and displays frames in less than a second, on average.

5.4 Xanadu

Project Xanadu [Nelson 80], as its author Theodor Nelson claims, is a system designed to be the principal publishing utility of the future.
Xanadu has been under continuous development for over 28 years. The program has been written in C under Unix. Its greater aspiration is a universal instantaneous hypertext publishing network.

The fundamental elements of the system are *documents* and *linkages*.

It is the different types of links that give both power and complication to the system. Certain link types, which have direct analogies in existing literature, are

1. The jump–link, corresponding to the footnote

2. The quote–link, corresponding to the printed quotation
3. The correlink, resembling the marginal note
4. The equilink, which is a link between the same thing in two versions of the same document

Many other types of links are allowed within the system. In principle it is allows any type of link to be defined by the sophisticated user.

A document in Xanadu can contain text, or links, or both. In this way everything in the system is a document and has an owner.

A main characteristic of the system is that only one copy of every document exists. Anything stored by one user in the system may be quoted by another person but no copy is made of the quoted materials.

Throughout the development of Xanadu two goals requirements have remained permanent: A user should be able to see and to follow arbitrary links between pieces of nonsequential text and to be able to compare different versions of documents.

5.5 Textnet

Textnet [Trigg 86] is a new system for structuring text. It has been designed in order to investigate novel strategies for text organization and their ramification for an on-line community. It is presently implemented only in the local level.
Some of the features that Textnet supports are advanced user interface, browsing and keyword searching, path establishment, general purpose chunk format, convenient critiquing[2], refereeing and relinking.

The Textnet network is organized using a semantic net formalism of labeled links and nodes. There are two types of nodes: *chunks* and *tocs* capturing the textual and hierarchical components respectively. Thus a chunk node represents a primitive piece of text whereas a toc node corresponds to an entry in a table of contents. Connections between the various nodes are made by typed (labeled) *links* The fourth data object in Textnet is the *path*, which is an ordered lists of nodes and which is used to browse linear concatenations of text from several nodes.

[2]By the term *critiquing* we mean that the system allows readers to attach critiques and other forms of commentary as new chunks onto existing nodes

There are four major activities performed by users accessing Textnet: text perusal, text and structure composition and modification, critiquing and reader linking, and hard-copy rendering.

5.6 Notecards

Notecards [Halasz 88] was developed at Xerox PARC using the Xerox Lisp environment. It is a general hypermedia environment which is fairly typical for the generation of workstation-based systems that are currently moving from the research lab into widespread use.
There are two primitive constructs in the system: *notecards* and *links*. In the basic system these two primitives are augmented by two specialized types of cards: *browsers* and *fileboxes*.
A notecard contains an arbitrary amount of editable substance such as pieces of text, structured drawings or bitmap images. A link is a typed, bidirectional connection between a source card and a destination card. The link type is a user-chosen label specifying the nature of the relationship being represented. A browser is a notecard that contains a structural diagram of a network of notecards. Fileboxes are specialized cards that can be used to organize or categorize large collections of notecards.
The information in Notecards can be accessed by browsing (navigation). A limited search facility is also provided.

5.7 Neptune

Neptune [Campdell 88] is a hypertext system for CAD applications. It was implemented by Computer Research Laboratory, and currently running on Unix 4.2.BSD.
Neptune is designed as a layered architecture. The bottom layer is a transaction based server called the Hypertext Abstract Machine (HAM).

The HAM presents a generic hypertext model which provides storage and access mechanisms for nodes and links. It provides distributed access over a computer network, synchronization for multi user access and transaction-based crash recovery.

Additionally, application layers are built on top of the HAM and a user interface is built on top of the application layer. The application layers consist of programs that automatically manipulate or transform hypertext

data.
The HAM storage model is based on five kinds of objects: graphs, contexts, nodes, links and attributes.
Ham supports an automatic version history mechanism. It also provides a filtering mechanism that allows subsets of HAM objects to be extracted from larger graphs.

HAM operations are grouped into seven categories:

- create operations
- delete operations
- destroy operations
- change operations
- get operations
- filter operations
- special operations (searching, merging, managing transactions)

Ham is designed as a general-purpose hypertext engine and can be used as a basis engine for other hypertext systems.

5.8 Document Examiner

Document Examiner [Walker 88] is a system implemented by Symbolics software and is one of the few commercially accepted Hypertext systems. It is used for the online manuals for the Symbolics Workstations and has two goals: To be a better tool for the technical writers and to allow easier access for customers. They set out to solve those two practical problems and not specifically to build a Hypertext system as such, even though that is what they ended up having!

They thought that readers should not have to work as hard as writers, so they designed two separate user interfaces to the system. The one for the writers is called *Concordia* while the *Document Examiner* is the reader's interface. Currently (1988) the system has 10.000 nodes and 23.000 explicit links between nodes as well as 100.000 implicit links. Each node has an average size of 1K, so the total system has 10M of info. The printed version has 8.000 pages.

The local engineering staff at Symbolics prefer the system over the paper version: Among 24 engineers surveyed [Walker 88], just two of them preferred the printed version!

5.9 Hyperties

Hyperties (Hypertext based on the Electronic Encyclopedia System) enables users to easily traverse a database of articles and pictures by pointing at highlighted words or images in context.

Hyperties has been under development since 1983 at the Human Computer Interaction Laboratory at the University of Maryland. It uses embedded menus rather than isolated and explicit menu items. Embedded menus are good examples of specialized indexing for systems that emphasize understanding rather than retrieval.

5.10 Electronic Encyclopedia

Electronic Encyclopedia system is an encyclopedia on CD–ROM. The printed version consists of 20 volumes. The hypertext version consists of 60 megabytes of text and 50 megabytes of indexes that contain pointers to each occurrence of every word in the encyclopedia. The powerful search software for this system provides rapid access to all occurrences of any word or phrase entered by the user.

5.11 Hypertext for Micros

Hypertext systems for micros are systems derived from the academic ones, for commercial use. Their main characteristics are that they have many limitations[3], they are cheap and they are popular.

5.11.1 Guide

Guide [BYTE 88], which runs on both the Macintosh systems and the IBM PC AT and compatibles running Microsoft windows, is a general–purpose

[3]There were many discussions on whether Hypercard was or was not *really hypertext* because of its limited possibilities for associated links with words [Nielsen 87].

hypertext tool. Some of the applications you can develop with it include on-line documentation, storyboards, E-mail, and CAI courseware.

Guide lets you create dynamic layered documents. The "guideline" network is organized in both a hierarchical and a nonhierarchical manner. To move about hierarchically, you use the *replacement* buttons, which follow embedded menus. You can also use *note* buttons to bring up complementary information, such as a definition of a word or phrase. An *inquiry* button, which reveals the other buttons at your disposal, is also available.

To follow nonhierarchical links, you use the *reference* button, which will jump you to a new document or a different section of the document you are in.

Guide 2.0 uses an internal script language to let you execute external programs from your Guide document. You can also access and control videodisk players and modems via the serial port.

Lastly, a version called CD-Guide lets you create CD-ROM applications.

5.11.2 HyperCard

HyperCard [BYTE 88], available for the Apple Mac II, the Mac Plus, and the Mac SE, is a personal organization tool and a simple database manager. It is also a commercial software developer's tool and is in use in some corporations as a frontend to the mainframe database.

This system uses screen sized cards (or window sized cards in Mac II) organized into topic related stacks to create simple databases. One card is displayed at a time. Touching your mouse cursor to a button on a card executes a script written in HyperTalk, HyperCard's programming language.

You can browse through already-created stacks (stackware), paint and type, author new cards and stacks, and write and edit HyperTalk scripts. (It is fairly easy to write scripts with HyperTalk because of its English-like syntax.)

HyperCard applications have been developed in many areas. Much stackware is available in the public domain.

6 General comments on Hypertext systems

6.1 Classifying Hypertext systems historically

Historically we can divide Hypertext[4] systems in two categories:
1) The first generation hypertext systems

- *NLS/Augment, FRESS, ZOG,*

which were mainframe based, focused primarily on text nodes and used display technologies with little or no graphics capabilities.
However they included at least some support for large teams of workers, sharing a common hypertext network.
2) The second generation hypertext systems (until early 1980s)

- *Notecards, Neptune, Intermedia, KMS,*

use improved workstation technology which led to much more advanced user interfaces. In particular all these systems support graphics and even animation nodes. They also make heavy use of graphical overviews of the network structure to aid navigational access. These systems are generally designed for single users or small workgroups and hence do not support collaborative work to the same degree as the earlier systems.

These workstation based, research oriented systems have been followed a few years later by a number of products and prototype systems running on personal computers. The PC-based systems (Guide, Hyperties, Hypercard) are of course more limited in scope and functionality than the earlier ones.

An important remark on all the above systems:

> Generally in these systems information access and structure editing is accomplished by using browsers containing a graphical map of the network structure. Query–based access is possible but is very slow and limited to simple string or keyword matching.

[4]In this paper we use the words Hypertext and Hypermedia as if they have the same meaning, unless something else is explicitly mentioned.

6.2 Further dimensions for classifying Hypertext systems

6.2.1 Scope

Hypermedia has been proposed as the mechanism for storing and distributing the world's entire literature output, as a common basis for teams of programmers on large software projects, and as a tool for individuals and small working groups engaged in authoring and idea processing. We add to these the single user hypermedia systems that have appeared in the last few years, and we have the following classification:

- **Single User**: Guide, Hypercard, Hyperties.
- **Working group**: Intermedia, NoteCards.
- **Corporate division**: Augment, ZOG.
- **Whole World**: Xanadu.

6.2.2 Browsing vs. Authoring

In systems designed primarily for *browsing*, the hypermedia network is carefully created by a relatively small number of specialized authors, in order to provide an information space to be explored by a large number of more or less casual users. These systems are characterized by relatively well-developed tools for information presentation and exploratory browsing.

In systems designed primarily for *authoring*, the hypermedia network serves as an information structure that is created and continuously modified by many authors. In such systems, the tools for creation and modification of the network are well developed. Tools for easy browsing and sophisticated information display are present but tend to be less evolved.

- **Focus on information presentation**: Intermedia, Document Examiner, Hyperties.
- **Focus on network creation and manipulation** : Augment, Neptune, Notecards, Xanadu.

6.2.3 Task orientation

Many Hypermedia systems have been designed to support a specific task. Other provide general hypermedia facilities to be used in a variety of applications.

- **General**: Guide, Hypercard, Neptune, ZOG, KMS, Xanadu, Textnet.
- **Some task orientation**: Augment, NoteCards, InterMedia.
- **Task specific**: Document Examiner, Electronic Encyclopedia.

7 Online compact journal

Some years ago a system aiming towards electronic publishing was implemented at the Katholic University of Nijmegen, The Netherlands (KUN). It was called *Online Compact–Journal* [Leo 84]. The main feature of this system is that the text is not organized in a linear fashion. Instead, it is organized as a hierarchical structure, similar to the directory structure in the Unix operating system. There are directories, subdirectories etc. The directories contain text split into pieces, called *compacts*. The directories are used for two purposes: for grouping compacts of the same subject, and then they are called *sections*, and for grouping other directories. A compact may belong to more than one section.

There are two kinds of users: *superusers*, who can create directories and compacts, and *users*, who can only navigate through the directories and read compacts. A user entering the system starts at the root directory. Then he can decide to move down the hierarchy guided by the contents of the subdirectories, or *search* for compacts with a specified *regular expression*. At any time he can ask the system to show his current position in the hierarchy.

The system was implemented with the idea of operating on local or wide area networks.

The idea of the hierarchical structure, together with the browsing and navigating capabilities, was a step towards a hypertext system, and this particular system has some desirable features, but some serious drawbacks too.

First of all, the hierarchical structure seems natural when you want to store text that is structured in the well known way: paper (or book), chapters, sections, subsections, paragraphs. The pattern matching facility is also a very powerful feature for such a system. The user can search for a pattern in the entire database (if he starts searching when he is in the root directory) or in a specific subtree (when he is in a subdirectory or section). The system offers also the user the possibility to see at any time his position in the hierarchy. Information is stored in the compacts in *named fields*, with names like "Title:", "Author:" and "Abstract:". This way the user can narrow and widen his view, deciding at any time what fields he wants to see or search. So, the naming scheme makes the hierarchy not only a way to describe the structure of documents, but primarily a *semantical structure*. A "Name:" field in every compact gives a meaningful name to it. These names serve as a guide to users, to help them in browsing and navigating through the database.

As we said before, the system had also some major drawbacks. First of all, there was a clear distinction between directories (that could only contain subdirectories or sections), sections (that could only contain compacts) and compacts (that contain the text). In other words, the system was rather inflexible. Also the system was only used for storing abstracts of papers and short news. No compact or a collection of compacts was a hole document. The navigation sometimes is rather difficult, if you are in a compact and decide to view another compact in a different directory.

Part II
Requirements and description of an Electronic Publishing system

8 Description of the system

8.1 Defining our aims

Until now, we have given a brief overview of many Electronic Publishing and Hypertext systems. They have a lot of common features and a lot of differences, too. The source of the main differences can be found in their task orientation. Some systems were built for document authoring and others for document reading and for retrieving information.

> Our main focus is on a system for *document reading.* This means a system which is mainly designed for casual readers who want to navigate through the documents database, read, find facts, link and annotate literature of their own interest.

Readers are used to searching and reading normal printed documents. Tables of contents, indices and references are well known ways for searching in literature. Also, readers are familiar with making annotations and adding comments to printed documents.

But printed documents are constrained by the medium. Parallelism of presentation, audiovisualization, quick updating of the information, direct following of references, are some of the features that should be offered by an electronic document system, using the power and the flexibility of the computer. Only then readers are going to change their habits and prefer this over printed media.

An electronic publishing system consists of two major parts, a *database part* and a *user interface part.*

> Our focus will be more on the database part and the functionality that it should provide and less on the user interface.

In the rest of this chapter we describe the structure of the database and the objects that constitute it. In the following chapter we will see how this structure can be used to provide some attractive features and fulfill the user's needs.

8.2 Structure of information and connectivity

8.2.1 Nodes, Links, Connections everywhere

The structure of information in such a system is much like the structure of hypertext systems. This means, first of all, that its main characteristic is *non-linear* or *non-sequential* text. Documents consist of a collection of **nodes**, called *cards, chunks* etc. in various systems, connected by **links**. A node is similar to a piece of normal text. The links provide hypertext with its nonlinear aspects. Links can connect nodes in the same or different documents. The complete collection of documents in a hypertext system can be thought of as one big *hyperdocument.* If the nodes and links of a hyperdocument are mapped to nodes and edges of an abstract graph, then a hyperdocument can map into a graph called the *hypergraph*[5].

In the general hypertext model there are no constraints on the kind of nodes and links in the database. The model is quite general and flexible. As we said earlier, our main aim is on storing documents (books, articles) and facilitating the process of document reading. We believe that the generality of this structure tends to pose difficulties in this process. So, some restrictions should be put, at least in the kind of links and the way they are created.

Because most documents have a hierarchical structure (chapters, sections, paragraphs), it is natural to assume that **hierarchical links** should be the first kind of links existing in such a graph. Therefore some nodes could serve for grouping nodes that constitute a section, a chapter etc. We also want to use hierarchical links to group documents on the same subject. It is usual for some document to refer to more than one subjects. So, we would like to have the ability to place the same node as a child of more than one nodes. In this way the hierarchical links do not implement a tree, but rather a *directed oriented acyclic graph (DOAG).* This means that two nodes can have the same descendant and a node can have more than one ancestor. From now on we are going to call the hierarchical links **connections**. This

[5]This definition of hypergraph has nothing to do with hypergraph definition found in graph theory.

way of structuring text is well known to the user, and he will not have great problems in understanding and accepting it.

Apart from the *connections* there will be **horizontal links** for connecting documents and nodes having some kind of not hierarchical relation (reference, annotation etc.). For instance, from the node of a paper containing the references of the paper there should be links to all the documents that the paper references. Looking at the nodes that reference a certain node will be possible, using these links. Or from a node there should be a link to another node that a user has added, annotating the contents of the first node. From now on we are going to call horizontal links just **links**.

After this short description someone can easily see that the three main objects in the database (fig. 2) are:

- **Nodes**, containing the actual information. From now on we will call them *compacts*, due to the origin of this work in the *online compact journal*,

- **Connections**, that construct the hierarchical structure of the database, that is a DOAG, and.

- **Links**, that are used for representing relations between compacts.

8.2.2 Compacts

The structure of a *compact* (a node in the database), is the following:

Name A unique ID given by the system to every compact.

List of in-connections List of the connections that enter the compact.

List of out-connections List of the connections that leave the compact.

List of in-links List of the links entering the compact.

List of out-links List of the links leaving the compact.

List of named fields The information in a compact is stored as a collection of named fields. The name of the field can be a string, while its value can be a piece of text, a picture or a number. The notion of fields serves two purposes.

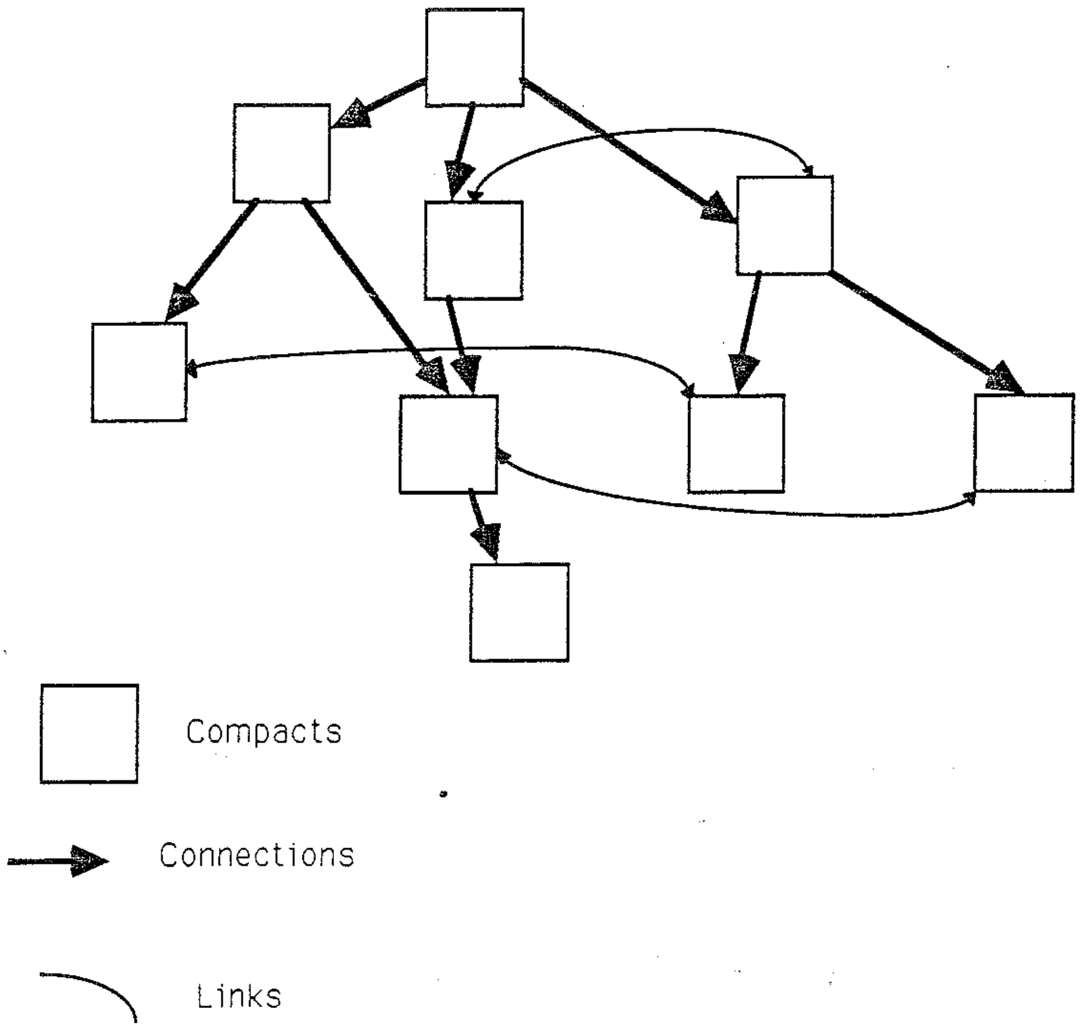

Figure 2: Objects and Structure of the Database.

First, fields are used for carrying the actual information (text, figures) of the compact and then the name of the field can be *Text:*, *Abstract:*, *Figure:*, *Example:* etc.
Second, they can be used for storing attributes of the compact. Then the name of the field can be *Author:*, *Subject:*, *Keywords:*, *DayOfCreation:*, etc.

8.2.3 Connections

Connections do not carry any special kind of information. They just contain the identifier of the compact that is the source of the connection *(father)* and the compact that is the destination of the connection *(child)*.

8.2.4 Links

The structure of a link is the following:

Name A unique ID given by the system to every link.

Source The source point of the link inside a compact. This point can be a single word, a sentence in the compact or even the whole compact. It can be described by three values:

1. The name of the compact,
2. The start position inside the text of the compact, and
3. The length of the piece of text that this link is leaving from.

If you just give the name of the compact without any other information that means that the start of the link is the whole compact.

Destination The ending point of the link in another compact. This position can be described by the same values as the source point.

List of named fields The same as the fields in a compact. This time they can be used for describing the type of the compact (reference, annotation, explanation, etc.) and even for giving a short description about the destination compact. With this information displayed users could decide if they really want to follow a link or not.

The structure of the nodes and the links are intentionally left quite general. In this way the system has greater flexibility. The fields associated with every compact and link can be user specified and their value is usually a string. They are a way of adding more semantic information to the hypergraph.

As we have said a major notion in the *online compact journal* (described earlier) was that of **fields**. Users could decide which fields they want to search, display or print. This mechanism is preserved and extended in the new system. Fields are a convenient way for representing both the value and the attributes of compacts. We also wanted to have a uniform way of representing compacts, connections and links, and fields as a mean of representation satisfy this requirement, too. So, connections leaving a compact Bcan be represented as numbered fields, and links leaving a compact can be represented as named fields, with the name being the type of the link (for instance, *Reference:*).

From the user's point of view, every compact can be seen as a collection of fields: some numbered fields, that have as values connections, some named fields (*Author:*, *Abstract:*) that have as value a text and some named fields (*Reference:*, *Comment:*) that have as values links leaving the compact. For the moment, the system puts no restriction and makes no check on the names of the fields. It is up to the user to assign meaningful names and values.

8.3 Some more features

Unfortunately, compacts and links although they offer ways of structuring information, are not enough. Sometimes the network can become very complicated and users can easily get lost during navigation. We believe that providing a map is a main responsibility of the user interface, but the database must also provide some facilities for helping users in such situations. Such facilities, which can also be used for other purposes, are:

Paths By *paths* we mean ordered sequences of compacts. The paths can be specified by the author or can be created by the user during a session. Paths are a part of the database. They also have a name and users can at any time select a path to follow, create a new path to be followed later by other users, or add links and compacts to a path. The system also provides a default path, that is created by the depth-first traversal of the graph. An easy way to describe paths is using fields that have links as values. So a *Next[name]* field could specify the next compact in the path, where name describes the name of the path (for example NextTutorial:).

Filters By *filtering* we mean a way of broadening and widening the view on the database. Users can do this in two ways. First, by specifying the fields that they want to display or search. Second, by specifying predicates that should hold for values of certain compact or link fields. In this way only a part of the database that satisfies the predicates should be selected. The first way of filtering is applied on the compact level every time only a part of the compact is displayed or searched. The second way is applied on the database as a whole, and only a part of the database is selected and accessed.

History The system always keeps track of the users movements in the database. So users can follow their steps and return to a known compact or save the history when they exit the system and return to the same position when they enter the system the next time.

In the next chapter we describe in greater detail how the structure of the database and the facilities it offers can be used .

9 Main features of the system

In the previous chapter we outlined the structure of the system. In this chapter we will try to justify our decisions and describe how the system can be used effectively and cleverly.

9.1 Why Connections (Hierarchical Links)?

The hierarchical organization of documents (articles and books) is well known to the users. Every document consists of chapters, chapters consist of sections, sections of subsections and paragraphs. Document processing systems like LATEXuse the keywords chapter, section, subsection, paragraph to let the writer describe the structure of the document. Compacts, connections and attributes associated with compacts[6] can be used to represent that structure. A field with name *Level* should be associated with every compact. It could have the values *Group, Document, Chapter, Section, Subsection and Paragraph.*

Compacts of level *Paragraph* should be used for actually storing pieces of text of the document. A question that arises here is "what should be the size of the text in such a compact?". The database allows any compact to have an arbitrary size. But we believe that each compact should contain text describing only a single "idea". Maybe this could be the size of a normal paragraph in a paper, or more. It is up to the creators of the documents to decide that size. If they discover later that too many links enter or leave the compact they could split it into smaller pieces, so that they can help readers in deciding what links to follow.

Compacts of level *Subsection* could be used for grouping compacts of level *Paragraph*, using connections. In the same way compacts of level *Section* could group *Subsections*, compacts of level *Chapter* group *Sections* and finally *Document* group all the sections of the paper. Compacts of type *Group* could group *Documents* of the same subject. All these do not mean that, for instance Chapter compacts should only contain Section compacts. The system does not impose such a restriction and it is up to the users to structure the document.

The main purpose of connections is to represent the relation *contents-of* between compacts. Users that look at a Document compact can see its

[6]As we said earlier, we can associate attributes with compacts, using the fields

contents by looking at the connections leaving the compact. The DOAG structure of the hierarchy is an important point. It is quite usual that you can find the same paragraphs or even the same sections in different documents written by the same user (or even by different users!).

The text in an intermediate compact (Chapter, Section, etc.) could be empty or there could be just a "Title:" field and a "Readme:" field that could contain a summary of the Chapter or Section.

On the other hand, hierarchy is a well-known way of representation. So, we believe that having the hierarchical structure as the default structure in the database is a first step towards helping the user staying oriented in the hypergraph. Obviously, it is easier to navigate in a tree-like structure, than in a general graph.

9.2 Why Links (Horizontal Links)?

Hierarchical links seem a natural decision. But why should someone use horizontal links? *Horizontal links are that which give the system its special character.* Hierarchical links (hierarchical structure) is just a way of storing documents in the computer but horizontal links are what make the difference between the way we look at a normal document and a hyperdocument.

Let us see some examples of how attributes on compacts and links in combination with horizontal links can be used.

- *References.* An obvious way that horizontal links can be used. When a document references another document there could be a link from the compact where there is the reference to the Document compact of the referenced document or even to a special point inside the document[7]. There should be a field with name *Type* and value *Reference* to describe the purpose of that link. So users can easily follow the references at any time, depending on their interests. They do not have to go up the whole hierarchy and start searching for the referenced compact, they can go to that with just one movement.

- *Footnotes.* Links can be used in the same way as for references for giving footnotes.

[7]This implies that the referenced document should exist in the same database, but we do not worry about that, at least for the moment.

- *Examples.* A link from a point in a document can lead to an example on an argument made in that compact. Actually, you can have a whole list of examples connected with links of type Example. The users can follow the links as far as they like or until they understand the argument completely. Compacts containing examples could have an attribute *Description* having the value *Example* associated with them. In that way there will be a distinction between compacts containing arguments and examples on them. You can take it even further, and classify the examples as *Simple, Advanced etc.* As we have said earlier there are no prespecified attributes and values. It is all up to the creator of the database and the way the user or an application (mainly the user interface) interprets these attributes.

- *Theorems and Proofs.* About the same things as for the examples can be also applied to the theorems and the proofs of a document. There could be a special attribute describing that a compact contains a *Theorem* or the *Proof* of a theorem. The theorem and its proof can be linked with a special link of type *Proof* and the users can decide to follow this link if they are curious or patient enough.

- *Commenting and Annotating.* All the examples we have seen until now deal with the structuring of the document from the author's point of view. Readers are used in making comments on what they read or underlining things that they think are important for them. The usual way to do this in a printed document is by adding comments in the margin. An electronic document should also offer that ability to the users. That can be done with special horizontal links of type *Comment* or *Annotation* that could lead to compacts containing readers comments. These compacts can be easily added while the users are reading a document. Comments can be added to a Paragraph compact inside a document, to an intermediate compact or even to the Document compact, if it is a review of the entire document. The ability of easily adding comments and easily finding all the comments about a document or an argument inside a document, should be offered as a main feature to an Electronic Publishing system. If the database is accessed by many users then a whole discussion can start around a document. Imagine that there could also be comments on comments, and all the feedback can be easily available to the author of the document.

- *Arguments*
- *Explanations*
- *More Information* All of these can be found in a printed document as a part of the whole document. In a system supporting non–linear writing and reading only the main thoughts in a document should be written linearly. All others could be described with links, associating a part of the main document with arguments on a position, explanations of a subject and more information on certain points. It is difficult, of course, for authors to start thinking and writing this way. But we have to make this step in moving from the printed media to the electronic book.

In the description for the hierarchical links we saw that they are a way for describing the structure of the documents. Now, using horizontal fields together with meaningful names attached to compacts we can add semantics on the hypergraph, something much more important than just structure. Of course, it is up to the user–writer to decide what are really meaningful names, and this is one of the greatest problems of non–linear writing.

9.3 Thinking about the user

After all we have said it is time to see how users can move inside all this complex structure and how they can use it so that they can effectively read a hyperdocument.

9.3.1 Browsing

As in all systems like this, **browsing** (or *navigating*) is the main way for looking and finding facts in the database. That means that at any time one compact is the *focus* or *current* compact that users are looking at and they can move by

1. following a link that leaves the current compact, or
2. going to a compact that they select from the contents of the current compact (actually following a connection if they are in an intermediate compact).

So, through navigation readers can read documents that are of interest to them, find associated pictures, examples, comments, etc.

When readers start a session they look at the *Root* compact of the database. That is the compact that groups all the subject compacts. He can choose to go to a specific subject and choose there again a specific document looking at a compact's children or move horizontally to other relevant points.

As we said earlier, the system is based on a hierarchical structure. This way the system supports *hierarchical browsing*, as it is defined in [Glushko 89]. By hierarchical browsing we mean exploration by the progressive display of detail. Users starting from the root of the database can move to a subject compact, then to a document compact (containing probably the table of contents for the document) and then keep moving to the leaves of the tree, to compacts describing something in greater detail.

So, the system supports both undirected, general browsing (through the use of links) and hierarchical browsing (through the use of connections).

9.3.2 Following a linear path

As we said in previous sections *Paths* (sequences of compacts) is an important feature of the database. Paths can be created by authors after they have written a document. They have a name and they include compacts that have some special property, are dealing with a special subject and should be read in order. So, when readers enter the database they can choose from a collection of paths one that addresses their needs and interests and follow it instead of navigating through the database and trying to find facts and compacts of interest by themselves. Paths can also be created by users if they find an interesting sequence of compacts that they or other users would like to follow later. By creating paths users can make their own "dynamic" documents. Paths can also be used for creating hardcopies of the database.

When users are following a path it does not mean that they can not follow another link and start navigating. After they stop navigating they can come back to the point they left the path and start following it again.

9.3.3 Searching

In a large database with thousands of compacts users can have troubles locating information solely by browsing, either because the structure of the database is not well suited to their search, or because they have forgotten where they or someone else have stored something. So, the system should offer a kind of **query language.** This query language can be very primitive and simple, like Boolean term searching or pattern matching, or more sophisticated. We think that some issues from automatic indexing would suit in this situation. Every document could have some keywords and there would be a kind of indexing on them. Anyway, some compacts should be selected and if a lot of them belong to the same document, then the corresponding Document compact should be selected. Then, these compacts should be presented to users with a hint of what is more relevant to their query.

The system we built supports pattern matching on substrings, contained in the fields of compacts. Users can specify a list of search fields. Then, only these fields are searched for the existence of a specified pattern. Users can start searching from the root of a subtree, giving a regular expression to be matched. All the compacts in this subtree are searched. At the end of the search a list of the compacts, that contain the pattern in the specified search fields, is returned.

We must mention here that in most of today's existing systems query based access is not implemented at all. When it is implemented it is very slow and limited to simple string or keyword matching.

9.3.4 Creating links and modifying the database

Of course, users can at any time add their own compacts (comments and annotations) and links and become *active* users instead of staying *passive.* Users can add a link to a compact they have created as a comment to an existing compact, or add a link connecting two existing compact that they believe they are relevant. In that way they can augment both the contents and the structure of the database. They can also create their own paths and update them as they navigate through the database.

Creating links and easily modifying the structure of the database is one of the strongest points of the hypertext model. Users can create their own webs of information and reuse existing compacts. New documents can be created just by adding links and connecting already existing pieces of text.

9.4 Addressing the needs of different users and different needs of users

With all these links and compacts the database can finally become too large and too complicated. If a lot of different users are accessing it a lot of new comments will be added and probably new papers or even books. The users who are going to use it will also have different interests, levels of education and experience in using the system. There will be users that enter the database for first time and others that have used it for a long time and know the system and the structure of the database very well. Another possibility is that the needs of the same user can vary from time to time. For instance, a user may want to find something quickly because he is in a hurry and he is looking for a clearly specified piece of information. Some other time the same user may just want to move around the database, have a look at the documents, without looking for something special (don't we often do the same when we enter a library or a book store and we are looking at a new book?).

So, there must be a way for fitting the services to the user (*customizability*). There are some things that the system can offer in this direction.

9.4.1 Filtering

As we have said, fields can describe attributes associated with every node and link. Using these attributes users can specify a *filter*, and select only those compacts and links between them that satisfy a set of predicates. In this way users can concentrate their attention on one aspect or part of the database (fig. 3). Of course, a special selection of attributes is needed, based on the contents of the compacts and the relation between them. For instance, there should be a way to filter out all the comments, or all the proofs of theorems.

Another way of widening and narrowing the view is the selection on fields. Users can at any time specify which fields of a compact they want to display or search. In this way, only a part of every compact is selected (fig. 4). So, they can just display the titles of compacts and the comments links. Filtering on attributes and selection of fields both serve the same purpose: widening or narrowing the users view. Their main difference is that by filtering on attributes you can only select a part of the database, while by selecting fields you can decide which fields of the compacts you want to see or search.

The filter notion is the same as the view notion found in relational

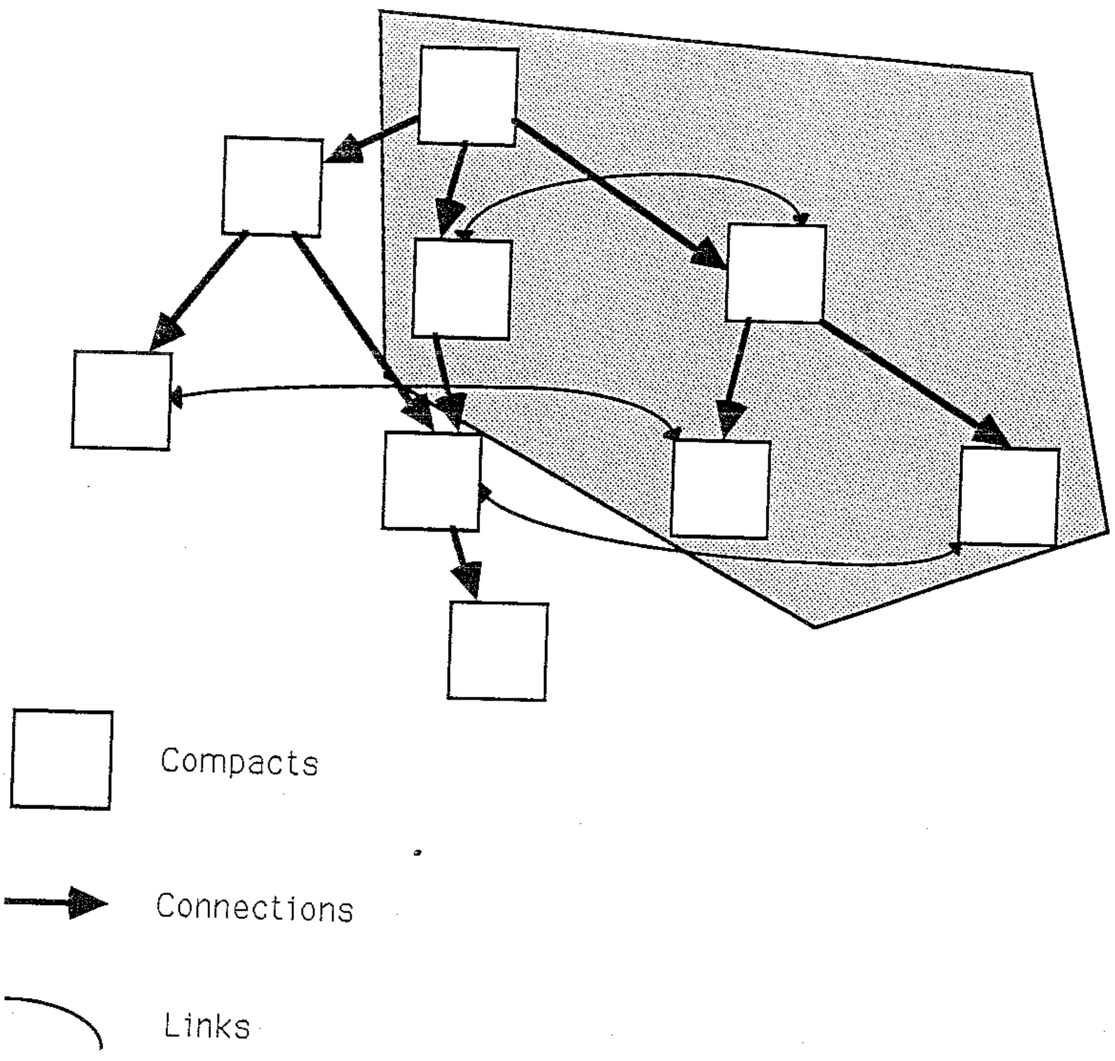

Figure 3: Filtering on attributes, a part of the database selected.

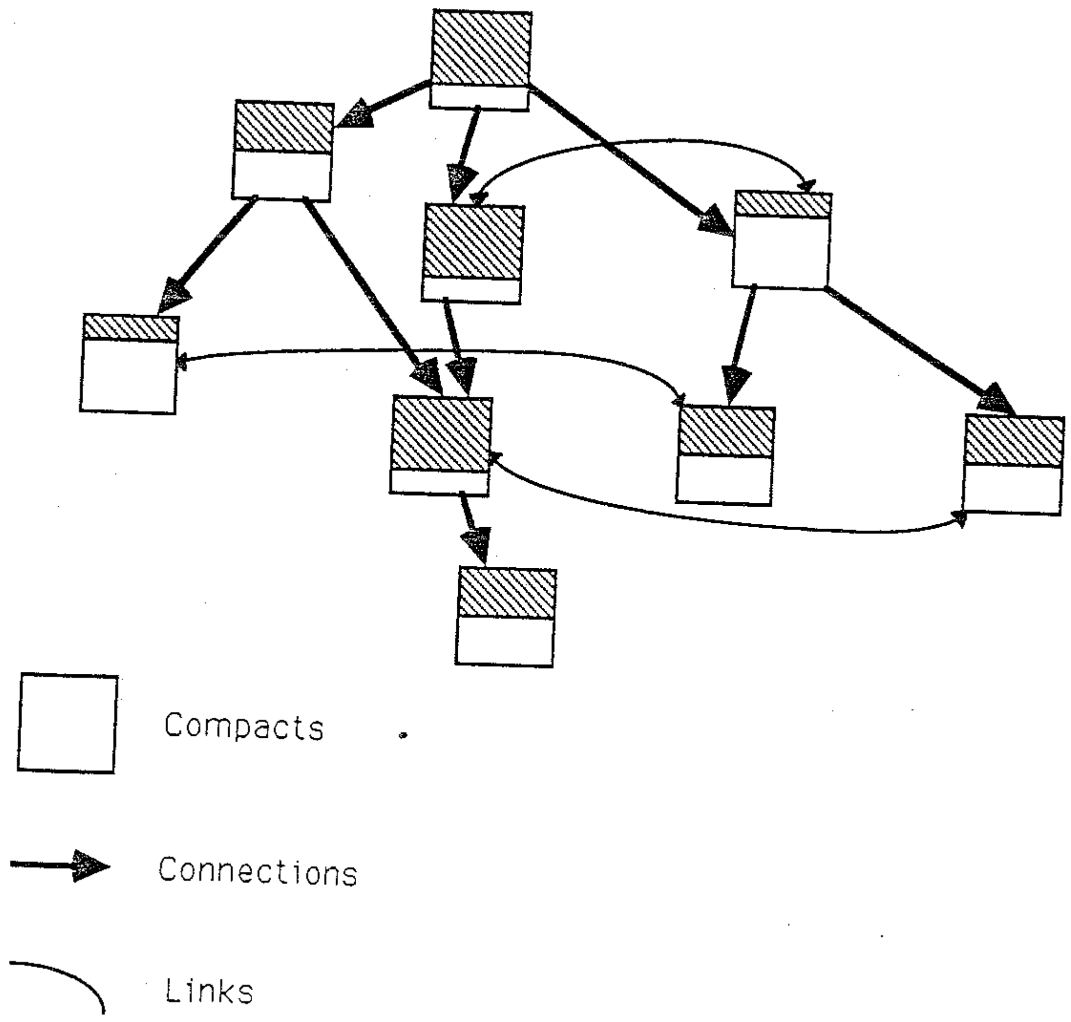

Figure 4: Filtering on fields, a part of every compact selected.

databases. Only a small part of the database is given to the users who may not be aware at all for the existence of other parts.

9.4.2 Using different paths for different users

The notion of *Path* can be also used for addressing users with different needs. For instance, if a book about databases is held in the system, a teacher may create a different path for an undergraduate course (selecting special sections, examples and exercises) and obviously he will give a path consisting of the whole book, and even some references to papers, to the postgraduate students. That way different documents can be created from the same material. This does not mean that students following the introductory course can not follow links leading to advanced material, if they want to do so. But at least if they follow the prespecified path they know what they *should* read and they will avoid reading confusing and useless material.

9.5 And if someone gets lost?

It is possible for users (even if they are experienced) to get lost in a complex network of information. We believe that it is mostly the task of the user interface to help users find their way (if it is quite advanced it could even offer a graphical map of the database and the current position of the user). Anyway, the database can also offer some help in this situation.

First of all, *filters* and *paths* are great aids. By filtering users can isolate a part of the database that they know well. Also, if they are following a path the possibilities of getting lost are greatly reduced.

The system always keeps a **history** of users' moves in the database. *History* is a system generated path and contains all the compacts and links that users have gone through. So, if they get lost they can follow the history and arrive to a known compact, or back to the compact they started navigating. Users should also have the capability of saving the history and the current status of the system before exiting. This way they can start a new session from the point they left the last one.

9.6 Some more words on links, fields and the need for a *Hypertext Administrator*

Until now we have seen that the database puts no restriction and make no classification about the users of the system. No distinction is made between users who can create new documents and others who can just read them. We have said that the main purpose of the system is to help the user–reader, so features like paths and filtering are implemented. We believe that is up to the system manager to classify the users and decide which of them should create documents, annotate documents or just read the contents. Attributes should be used to classify compacts and applications should use them to allow different operations for different classes of users. For instance, some users would only have privilege of creating compacts and links of type *Annotation*, so that they could only read the database and add comments. At any time later they would have the option of looking at all the comments, looking at just their comments or looking at no comments at all. Other users would have the privilege of creating and adding whole documents in the database.

Although the hypertext model is very simple (just nodes containing information and links connecting the nodes), the whole graph can eventually become too complicated because of a careless use of links. Also, the right selection of fields is an added overhead for a user of such a system.

Another issue arises now, that of the *Hypertext Administrator*. We know that in every database installed in any organization there is someone who is responsible for maintaining the database, writing new applications for it, creating views and providing different privilege of access for different classes of users. The hypertext system as described until now offers great flexibility and power, but sometimes this power can become useless or even catastrophic if someone can not use it in the right way. Imagine for instance that a user creates a filter that selects the entire database or just very few compacts. We think that there should be a clever assignment of the attributes and also a good set of filters that the users can use and have some meaning. So, an experienced user is needed. He should be experienced not only with the functionality of the database but also with the creation and editing of documents. That user should be responsible for the organization of the database, the selection of attributes and their values and even the writing of new applications using the underlying structure (for instance applications that take as input text in linear form and create documents in the hypergraph). Such work is not to be left to a simple user. One or more *Hypertext Administrators* or *Editors* are needed.

Another obvious problem is the maintenance of links. Of course, you can leave it up to the authors to decide what links they want to leave from their compacts. But if you want a link from every term to all the places that reference it (i.e. an index), this is rather difficult to do manually. A solution could be that with every compact should be associated a list of keywords. Then the automatic creation of all the links from compacts that reference such a keyword could be possible.

When talking about a general hypertext system, complexity and sometimes confusion can not easily be avoided. If every user can add links and create compacts, a “spaghetti” graph will be created very quickly. If on the other hand we are thinking of storing a certain book or a certain amount of articles in the database then some work has to be done from the beginning. We are talking about the *design of a hypertext database*. The designers have to decide how the text should be actually split in compacts so that redundancy could be avoided. Also, there will probably be a great deal of candidate links. The designers have to decide which links should be created, based on the kind of the book and the needs of the users that are going to read it. If the user's main task is to wander and explore the database, then unconstrained linking is not really a problem. But if the task is to access information and find facts quickly, then an analysis, identifying the text units and the links, should proceed the creation of the database. Because there is no great experience in the use of hypertext systems for big applications, no rules for the design of such systems have been established yet.

10 User interface considerations

As we have explained in the previous chapters our focus in this work was mostly on the conceptual database level. Nevertheless in this section we are giving our thoughts about how the user interface in Electronic Publishing systems should be.

10.1 General issues

User interface issues have always been a major focus in Electronic Publishing systems development. The userinterface is the yardstick by which the user judges the system. If a system has a poorly designed interface, it is liable to be rejected, irrespective of the facilities which it offers.

Improvements in human–computer interface design have come rapidly in the last decade. Hardware advances offer designers a range of input devices (such as mice or touch panels or even touch screens) and output devices (such as high resolution screens).

Likewise, software advances allow designers to choose from a variety of selection mechanisms in addition to traditional command languages. Although command languages offer expressive power to expert users, a variety of menu selection styles (pull-down menus, icons) offer novice and casual user ease of use. Instead of remembering commands or traversing menus, the user sees a representation of the "world in action". Because the user points (with mouse, touch screens etc.) at given objects and actions, the impact of actions is immediately visible, thereby reducing errors and speeding performance.

As with information retrieval, a key design problem is balancing the ease of learning afforded by direct manipulation and the access to powerful features for experts. The degree of user control provided must depend on the user; his or her purpose, and the task domain. Expert users who are specialists in the task domain will welcome great power and control, but novices to the system and the task domain will likely benefit from limited menus and less control.

There is a number of problems that we have to consider in building a user interface for an Electronic Publishing system.

10.1.1 How do we present information in an easily understandable way?

The most commonly used approach is that of having windows on the screen associated with nodes of the system. As the user views a node, visible links may be followed or not at the discretion of the reader. If a link is followed, then the node at the end of that link is made visible so that it may be read in turn.

10.1.2 How do we represent links?

There are different answers/solutions to this question that are highly depended on the hardware configuration someone is willing to use.

An easily implemented solution is to use **text highlighting** to present a link source on the screen (boldface, video reversing, underlining etc.).

Another solution is to have some special space on the screen (e.g. a window) dedicated for the links. In this way the links are isolated from the text.

10.1.3 How do we prevent users from getting lost?

The classical problem in such systems is the problem of *getting lost* (disorientation), that becomes more severe as the database grows larger.

The user interface is to a great extent responsible for helping the users *stay oriented* or *reorienting* themselves in case they get lost.

A helpful facility that a well–designed user interface should provide is a graphical representation of:

- the hierarchy
- the way followed till then
- the different choices from this position.

10.1.4 How do we help users find what they are looking for?

Users don't just need to know where they are, but also need to know how to find the information they are looking for.

The most obvious way of doing so, is by navigating around the database. But in large systems users can have problems searching for something by navigating alone. That's why such systems should provide a *search* facility. Using this facility users can search for text strings within the database.

10.1.5 How fast should the system respond when following a link?

The response time is another important factor influencing the user's opinion of the system.

We believe that fast system response when selecting a field is a very important parameter of Electronic Publishing systems. Although the average time a user spends at a node will usually be many seconds, there will be frequent bursts of rapid navigation, when response time becomes critical.

10.1.6 How do we help users?

The system should provide built–in *help facilities*. These should be accessible directly from the user and should provide different levels of help and advice; from very basic information on how to get started with the system up to full description of system facilities and how to use them.

11 Looking at the future

11.1 World–wide Electronic Publishing systems

The obvious next step to a system like this is a *network* version, a publishing network for anybody's documents, which users may combine and link freely. The system should be able to grow without size limits containing in the body of available writings whatever anyone has stored from any place on the network.

The need for world–wide standards specification arises. Only this way, databases or Electronic Publishing systems in different geographic latitudes can communicate. Unfortunately in ourdays the only thing standard about such systems is that there are no standards.

This *publishing network* will open the way to a new form of electronic publishing with many differences from the traditional (well known) publishing world.

But first there are some more "practical" problems to deal with in such a system:

- What about freedom of speech?
- How will the royalties be paid?
- What about copyright?
- What about private documents?
- What about integrity of the database?

11.2 Will there be just text?

The user of such a system will not change from his favorite magazine with its nice illustrated pictures unless he could find the same features in this system. That's why the system should offer more applications than just text manipulation.

The first important application is *image processing*. When the user reads something about classical Greek history and he wants to see what the "Parthenon" looks like, he should just have to click his mouse on the word "Parthenon" and an image of it should appear on its screen.

Another application that the system should support is *computer-generated sound*. That could help for example students on music. Integrating a music editor, a piano keyboard, and a synthesizer would allow students to *hear* every note they *see* as they write or play the notes.

There is a spectrum of media that can be included in E.P. systems. Static text, structured graphics, bit-map images, charts, and graphs fall at the low end of the spectrum. The inclusion of animation, computer-generated sound, and audio and video recordings add a richness to Electronic Publishing systems that is impossible to recreate with paper media.

11.3 CD–ROM distribution

A further market development is the rise in popularity of CD–ROM as a publishing medium. The acceptance of an industry–standard solution for CD–ROM (the High Sierra data format [Cook 88]), has made it possible for publishers to issue documents on optical disks [Glushko 89] (although not yet in a great extent), in the certainty that users can access the information using devices from a wide range of manufacturers.

Typical optical disk products currently available on the market are capable of storage on the order of *one gigabyte* of data on one surface of a disk platter [Christodoulakis 86]. The main advantages of these devices are:

- very low cost per bit of stored information (four to five orders of magnitude less than the cost of storage on magnetic disks.)
- long archival life (more than 10 years, versus the two to three years of storage provided by magnetic disks.)
- very high storage density (one order of magnitude higher than that provided by magnetic disks.)
- direct access capability.

On the other side of the balance, their access time remains problematical, but it will improve in the future.

We believe that in the future very large amounts of information will exist in Electronic Publishing or other systems within computers, and it appears to us that this information will be stored and distributed on *optical disks*.

12 Conclusions

Electronic Publishing systems mainly represent a new way of structuring text and information. This implies a new way of representing knowledge. As the advances of technology bring new powerful workstations, high–resolution displays and new mass–storage devices in everyday use, Electronic Publishing seems a promising approach to the creation of books and documents in non-linear form. This means that we have to change our way of looking at books and documents. Good use of Electronic Publishing features requires a proper understanding of the new medium.

Apart from this, there are other areas that need further research. First of all, the need for standardization in the format and the exchange of documents. There is a high degree of standardization in the area of printed documents, but only now some standardization work is done for electronic documents. Related to this is the distribution of hypertext databases using wide area networks. Finally, there is the need for the successful integration of different applications in the same framework. Different applications for creating text, graphics, sound or even animations that could all be in the same hypertext database. The most promising current technology for this seems to be that of object–oriented programming.

Something else that we have to consider is the new way of authoring. Until now writers where mostly concerned with organizing their thoughts in a linear form, one phrase after the other. If we want to have successful "electronic books", a great deal of research and work has to be done towards non–linear organization of text. Authors have to develop new skills in writing for electronic media.

From the writer's point of view, Electronic Publishing systems are the next generation of word processing. From the reader's point of view, they are a new generation of database management systems. Their importance lies in their capacity to augment and amplify human intellect. We need this capacity because of the exponential growth and increasing complexity of scientific, economic, medical, and other knowledge.

We believe that Electronic Publishing systems offer a viable alternative to all forms of reading, writing, archiving and study now handled by methods of paper.

References

[Akscyn 88] Akscyn, R.M., McCracken, D.L., and Yoder, E.A.
KMS: A distributed Hypermedia System for managing knowledge in organizations.
Communications of the ACM, July 1988, pp. 820–835

[BYTE 88] Hypertext.
BYTE, October 88

[Campdell 88] Campdell, B., and Coodman, J.M.
HAM : A General Purpose Hypertext Abstract Machine.
Communications of the ACM, July 1988, pp. 856–865

[Chen 88] Chen, P., and Harrisson, M.
Multiple representation document development.
IEEE Computer, January 88, pp. 16-25

[Christodoulakis 86] Christodoulakis, S., and Falutsos, C.
Design and performance considerations for an optical disk–based, multimedia object server.
IEEE Computer, December 1986

[Cook 88] Cook, P., and Williams, I.
Design issues in large hypertext systems for technical documentation.
Hypertext Conference. Aberdeen 1988

[Conklin 87] Conklin, J.
Hypertext: An introduction and Survey.
IEEE Computer, September 1987

[Delisle 86] Delisle, N., and Schwartz, M.
Neptune: a Hypertext System for CAD Applications.
ACM SIGMOD, May 1986

[Glushko 89] Glushko, R.J.
Transforming text into Hypertext for a compact disk encyclopedia.
CHI Proceedings, May 1989, pp. 293–298

[Halasz 88] Halasz, F.G.
Reflections On Notecards: Seven issues for the next generation of hypermedia systems.
Communications of the ACM, July 1988, pp. 836–852

[Hanson 87] Hanson, R.
Toward Hypertext Publishing.
SIGIR Forum

[Leo 84] Leo, J.M.I.M., Koster, C.H.A. and Zoethout, T.A.
The Ifip compact–journal, presented at IFIP word congress.
IFIP word congress, Tokyo, Japan, 1984

[Marchionini 88] Marchionini, G., and Shneiderman, B.
Finding Facts vs. Browsing Knowledge in Hypertext Systems.
IEEE Computer, January 88, pp. 70–79

[Nelson 80] Nelson, H.T.
Replacing the printed word: A complete literary system.
IFIP, 1980

[Nielsen 87] Nielsen, J.
Tripreport: HYPERTEXT '87.
SIGCHI Bulletin, April 88

[Rubinstein 88] Rubinstein, R.
Digital typography.
Addison–Wesley p.c. 1988

[Salton 88] Salton, G.
Another look at automatic text-retrieval Systems.
Communications of the ACM, July 1988

[Trigg 86] Trigg, R.H., and Weiser, M.
TEXTNET : A Network-Based Approach to Text Handling.
ACM Transactions on Office Information Systems, January 1986, pp. 1–23

[Walker 88] Walker, J.
Supporting document development with Concordia.
IEEE Computer, January 1988

[Yankelovich 85] Yankelovich, N., Meyrowitz, N., and VanDam, A.
Reading and Writing the Electronic Book.
IEEE Computer, October 1985, pp. 15–29

[Yankelovich 88] Yankelovich, N., Haam, B.S., Meyrowitz, N.K., and Drucker, S.M.
Intermedia: The Concept and the Construction of a Seamless Information Environment.
IEEE Computer, January 1988,pp. 81–96