Hypertext: An Introduction and Survey

Jeff Conklin

Microelectronics and Computer Technology Corp.

Most modern computer systems share a foundation which is built of directories containing files. The files consist of text which is composed of characters. The text that is stored within this hierarchy is linear. For much of our current way of doing business, this linear organization is sufficient. However, for more and more applications, a linear organization is not adequate. For example, the documentation of a computer program* is usually either squeezed into the margins of the program, in which case it is generally too terse to be useful, or it is interleaved with the text of the program, a practice which breaks up the flow of both program and documentation.

As workstations grow cheaper, more powerful, and more available, new possibilities emerge for extending the traditional notion of "flat" text files by allowing more complex organizations of the material. Mechanisms are being devised which allow direct machine-supported references from one textual chunk to another; new interfaces provide the user with the ability to interact directly with these chunks and to establish new relationships between them. These extensions of the traditional text fall under the general category of *hypertext* (also known as *nonlinear text*). Ted Nelson, one of the pioneers of hypertext, once defined it as "a combination of natural language text with the computer's capacity for interactive branching, or dynamic display . . . of a nonlinear text . . . which cannot be printed conveniently on a conventional page."[1]

Hypertext systems feature machine-supported links—both within and between documents—that open exciting new possibilities for using the computer as a communication and thinking tool.

This article is a survey of existing hypertext systems, their applications, and their design. It is both an introduction to the world of hypertext and, at a deeper cut, a survey of some of the most important design issues that go into fashioning a hypertext environment.

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 labelled tokens) and in the database (as pointers). (See Figure 1.)

But this simple idea is creating much excitement. Several universities have created laboratories for research on hypertext, many articles have been written about the concept just within the last year, and the Smithsonian Institute has created a demonstration laboratory to develop and display hypertext technologies. What is all the fuss about? Why are some people willing to make extravagant claims for hypertext, calling it "idea processing" and "the basis for global scientific literature"?

In this article I will attempt to get at the essence of hypertext. I will discuss its advantages and disadvantages. I will show that this new technology opens some very exciting possibilities, particularly for new uses of the computer as a communication and thinking tool. However, the reader who has not used hypertext should expect that at best he will gain a perception of hypertext as a collection of interesting features. Just as a description of electronic spreadsheets will not get across the real elegance of that tool, this article can only hint at the potentials of hypertext. In fact, one must work in current hypertext environments for a while for the collection of fea-

Documentation is the unexecutable English text which explains the logic of the program which it accompanies.

0018-9162/87/0900-0017$01.00 ©1987 IEEE

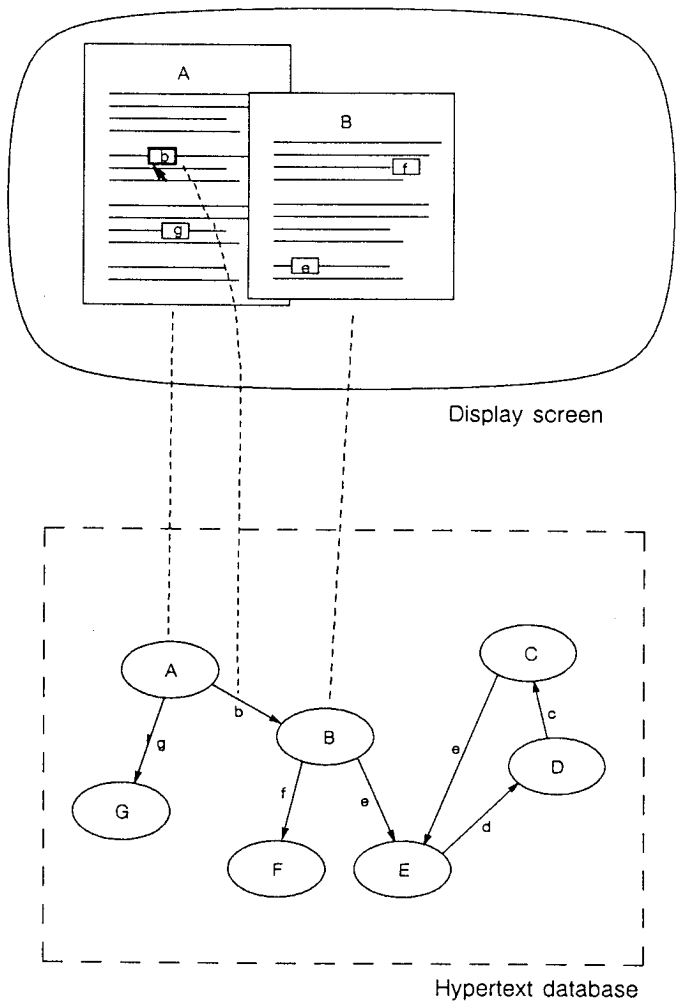


Figure 1. The correspondence between windows and links in the display, and nodes and links in the database. In this example, each node in the hypertext database is displayed in a separate window on the screen when requested. The link named "b" in window A has been activated by a pointing device, causing a new window named "B" to be created on the screen and filled with the text from node B in the database. (Generally, links can have names that are different from the name of the node they point to.)

tures to coalesce into a useful tool.

One problem with identifying the essential aspects of hypertext is that the term "hypertext" has been used quite loosely in the past 20 years for many different collections of features. Such tools as window systems, electronic mail, and teleconferencing share features with hypertext. This article focuses on machine-supported *links* (both within and between documents) as the essential feature of hypertext systems and treats other aspects as extensions of this basic concept.* It is this linking capability which allows a nonlinear organization of text. An additional feature that is common to many hypertext systems is the heavy use of windows that have a one-to-one correspondence with nodes in the database. I consider this feature to be of secondary importance.

One way to delimit hypertext is to point out what it is not. Briefly, several systems have some of the attributes of hypertext but do not qualify. Window systems fall into this category; while window systems do have some of the interface functionality, and therefore some of the "feel" of hypertext, window systems have no single underlying database, and therefore lack the database aspect of hypertext. File systems also do not qualify as hypertext; one could claim that a file system is a database, and that one moves among *nodes* (files) by simply invoking an editor with their names. However, to qualify as hypertext, a system must use a more sophisticated notion of links and must provide more machine support for its links than merely typing file names after a text editor prompt. Similarly, most outline processors (such as ThinkTank) do not qualify. They provide little or no support for references between outline entries, although their integrated hierarchical database and interface do approximate hypertext better than the other systems that I have mentioned. Text formatting systems (such as Troff and Scribe) do not qualify. They allow a tree of text fragments in separate files to be gathered into one large document; however, this structure is hierarchical and provides no interface for on-line navigation within the (essentially linear) document. Similarly, database management systems (DBMSs) have links of various kinds (for example, relational and object-oriented links), but lack the single coherent interface to the database which is the hallmark of hypertext.

As videodisc technology comes of age, there is growing interest in the extension of hypertext to the more general concept of *hypermedia*, in which the elements which are networked together can be text, graphics, digitized speech, audio recordings, pictures, animation, film clips, and presumably tastes, odors, and tactile sensations. At this point, little has been done to explore the design and engineering issues of these additional modalities, although many of the high-level design issues are likely to be shared with hypertext. Therefore, this survey will primarily address the more conservative text-based systems.

A glimpse of using hypertext. It is use-

*While this article seeks to establish the criterion of machine-supported links as the primary criterion of hypertext, this is by no means an accepted definition. Therefore I will also review and discuss some systems which have a weaker notion of links.

ful to have a sense of the central aspects of using a hypertext system, particularly if you have never seen one. Below is a list of the features of a somewhat idealized hypertext system. Some existing systems have more features than these, and some have fewer or different ones.

- The database is a network of textual (and perhaps graphical) nodes which can be thought of as a kind of *hyperdocument*.
- Windows on the screen correspond to nodes in the database on a one-to-one basis, and each has a name or title which is always displayed in the window. However, only a small number of nodes are ever "open" (as windows) on the screen at the same time.
- Standard window system operations are supported: Windows can be repositioned, resized, closed, and put aside as small window icons. The position and size of a window or icon (and perhaps also its color and shape) are cues to remembering the contents of the window. Closing a window causes the window to disappear after any changes that have been made are saved to the database node. Clicking with the mouse on the icon of a closed window causes the window to open instantly.
- Windows can contain any number of *link icons** which represent pointers to other nodes in the database. The link icon contains a short textual field which suggests the contents of the node it points to. Clicking on a link icon with the mouse causes the system to find the referenced node and to immediately open a new window for it on the screen.
- The user can easily create new nodes and new links to new nodes (for annotation, comment, elaboration, etc.) or to existing nodes (for establishing new connections).
- The database can be browsed in three ways: (1) by following links and opening windows successively to examine their contents, (2) by searching the network (or part of it) for some string,** keyword, or attribute value, and (3) by navigating around the hyperdocument using a *browser* that displays the network graphically. The user can select whether the nodes and links display their labels or not.

The browser is an important component of hypertext systems. As the hyperdocument grows more complex, it becomes distressingly easy for a user to become lost or disoriented. A browser displays some or all of the hyperdocument as a graph, providing an important measure of contextual and spatial cues to supplement the user's model of which nodes he is viewing and how they are related to each other and their neighbors in the graph. (See Figure 2.)

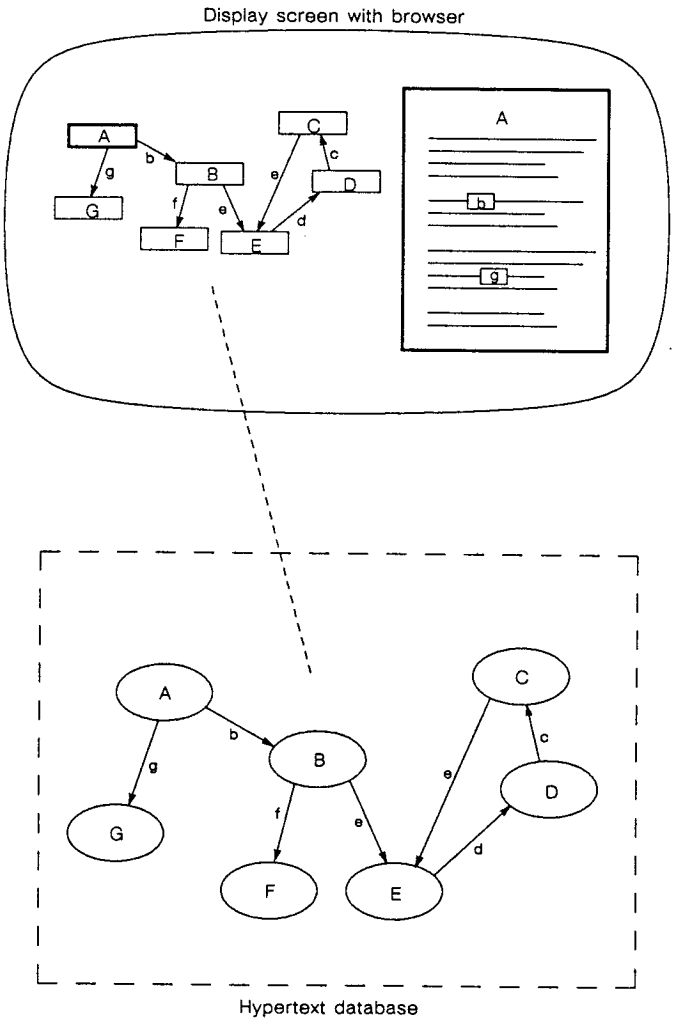


Figure 2. The screen at the top illustrates how a hypertext browser provides a direct two-dimensional graphic view of the underlying database. In this illustration, the node "A" has been selected for full display of its contents. Notice that in the browser view you can tell not only which nodes are linked to A but also how the subnetwork fits into the larger hyperdocument. (Of course, hyperdocuments of any size cannot be shown all at once in a browser—only portions can be displayed.)

Using a browser can be likened to using visual and tactile cues when looking for a certain page in a book. Sometimes we remember the general way the page looked and about how far it was through the book, although we don't recall the page number or even which keyword terms would help us find it by using the index or table of contents. The browser display can be similarly scanned and scrolled when the

*Note that I am are describing two uses of icons: those that function as placeholders for windows that have been temporarily put aside, and those within windows that represent links to other nodes.

**A *string* is a series of alphabetic and numeric characters of any length, for example "listening" or "G00274."

user has forgotten all but the appearance or location of a node.

Hypertext implementations

The history of hypertext is rich and varied because hypertext is not so much a new idea as an evolving conception of the possible applications of the computer. Many people have contributed to the idea, and each of them seems to have had something different in mind. In this section, I will review these theorists and their ideas in an effort to present a historical perspective as well as to sketch some of the hypertext applications that have been devised to date. I do not describe the individual systems and ideas reviewed here in any detail. For more detailed information, the reader is invited to consult the literature directly.

One kind of manual hypertext is the traditional use of 3×5 index cards for note taking. Note cards are often referenced to each other, as well as arranged hierarchically (for example, in a shoebox or in rubber-banded bundles). A particular advantage of note cards is that their small size modularizes the notes into small chunks. The user can easily reorganize a set of cards when new information suggests a restructuring of the notes. Of course, a problem with note cards is that the user can have difficulty finding a specific card if he has many of them.

Another kind of manual hypertext is the reference book, exemplified by the dictionary and the encyclopedia. In the sense that each of these can be viewed as a graph of textual nodes joined by referential links, they are very old forms of hypertext. As one reads an article or definition, explicit references to related items indicate where to get more information about those items. The majority of people's transactions with a dictionary make use of the linear (alphabetic) ordering of its elements (definitions) for accessing a desired element. An encyclopedia, on the other hand, can best be used to explore the local nodes in the "network," once one has found the desired entry through the alphabetic index.

There are also many documents in which references to other parts of the document, or to other documents, constitute a major portion of the work. Both the Talmud, with its heavy use of annotations and nested commentary, and Aristotle's writings, with their reliance on references to other sources, are ancient prototypes of hypertextual representation.

But if one insists, as most modern proponents of hypertext do, that navigation through hypertextual space must be computer-supported in order to qualify as true hypertext, then the field is narrowed considerably, and the history likewise shortened.

In some ways, the people who first described hypertext—Bush, Engelbart, Nelson—all had the same vision for hypertext as a path to ultimate human-computer interaction, a vision which is still alive today among hypertext researchers. Thus the historical review below stresses the early development of ideas about hypertext as much as the more contemporary implementation efforts.

Because of the difficulty of precisely classifying hypertext systems according to their features, my description will list systems according to application. There are four broad application areas for which hypertext systems have been developed:

- *macro literary systems*: the study of technologies to support large on-line libraries in which interdocument links are machine-supported (that is, all publishing, reading, collaboration, and criticism takes place within the network);
- *problem exploration tools*: tools to support early unstructured thinking on a problem when many disconnected ideas come to mind (for example, during early authoring and outlining, problem solving, and programming and design);
- *browsing systems*: systems similar to macro literary systems, but smaller in scale (for teaching, reference, and public information, where ease of use is crucial);
- *general hypertext technology*: general purpose systems designed to allow experimentation with a range of hypertext applications (for reading, writing, collaboration, etc.)

These categories are somewhat informal. Often the single application to which a system has been applied to date determines which category it is described in. Bear in mind that some of the systems mentioned below are full-scale environments, while others are still only conceptual sketches. Some systems have focused more on the development of the *front end* (the user interface aspects), while others have focused on the database issues of the *back end* (the database server). Table 1 identifies various features of the different hypertext systems which have been implemented.

Macro literary systems. The earliest visions of hypertext focus on the integration of colossal volumes of information to make them readily accessible via a simple and consistent interface. The whole network publishing system constitutes a dynamic corpus to be enriched by readers without defacing the original documents; thus, the difference between authors and readers is diminished. The advent of the computer has brought this vision closer to reality, but it has also revealed the monumental problems inherent in this application area.

Bush's Memex. Vannevar Bush, President Roosevelt's Science Advisor, is credited with first describing hypertext in his 1945 article "As We May Think,"[2] in which he calls for a major postwar effort to mechanize the scientific literature system. In the article, he introduces a machine for browsing and making notes in an extensive on-line text and graphics system. This *memex* contained a very large library as well as personal notes, photographs, and sketches. It had several screens and a facility for establishing a labelled link between any two points in the entire library. Although the article is remarkably foresightful, Bush did not anticipate the power of the digital computer; thus his memex uses microfilm and photocells to do its magic. But Bush did anticipate the information explosion and was motivated in developing his ideas by the need to support more natural forms of indexing and retrieval:

> The human mind . . . operates by association. Man cannot hope fully to duplicate this mental process artificially, but he certainly ought to be able to learn from it. One cannot hope to equal the speed and flexibility with which the mind follows an associative trail, but it should be possible to beat the mind decisively in regard to the permanence and clarity of the items resurrected from storage.[2]

Bush described the essential feature of the memex as the ability to tie two items together. The mechanism is complex, but clever. The user has two documents that he wishes to join into a *trail* he is building, each document in its own viewer; he taps in the name of the link, and that name appears in a code space at the bottom of

Table 1. Hypertext systems and their features.

Hypertext Systems	Hierarchy	Graph-based	Link Types	Attributes	Paths	Versions	Procedural Attachment	Keyword or String Search	Text Editor	Concurrent Multi-users	Pictures or Graphics	Graphical Browser
Boxer	Yes	Yes	Fixed[1]	No[1]	No	No	Yes	Yes	Emacs	No	Yes	Yes
CREF	Yes	Yes	Yes	No	No	By link	No	Yes	Zmacs	No	Yes	No
Emacs INFO	Yes	No	No	No	No	No	No	Yes	Emacs	No	No	No
IBIS	Yes	Yes	Yes	No	No	By link	No	No	A basic text editor	Yes	No	No
Intermedia	Yes	Yes	Yes	Yes	No[2]	No	No[2]	Yes	Custom	Yes	Yes	Yes
KMS	Multiple	Yes	Fixed	No	No[1]	Yes	Yes	Yes	Text/ graph. WYSIWYG	Yes	Yes	No
Neptune	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Smalltalk-80 editor	Yes	Yes	Yes
NLS/Augment	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Custom	Yes	Yes	No
NoteCards	Multiple	Yes	Yes	Nodes	No	No	Yes	Yes	Interlisp	Yes	Yes	Yes
Outline Processors	Yes	No	No	No	No	No	No	Yes	Various	No	No	No
PlaneText	Unix file sys.	Yes	No	No	No	No	No	Unix/ grep	SunView text ed.	Yes	Yes	Yes
Symbolics Document Examiner	Yes	Yes	No	No	Yes	No	No	Yes	None	No	No	No
SYNVIEW	Yes	No	No	No	No	No	No	No	line ed./ Unix	No	No	No
Textnet	Multiple	Yes	Yes	Yes	Yes	No	No	Keyword	Any	No	No	No
Hyperties	No	Yes	No	No	No	No	No	No[2]	A basic text editor	No	Yes	No
WE	Yes	Yes	No	Fixed	No[2]	No[2]	No[2]	No	Smalltalk-80 editor	No[2]	Yes	Yes
Xanadu	No	Yes	Yes	Yes	Yes	Yes	No	No	Any	No	Yes	No
ZOG	Yes	No	No	No	No	No	Yes	Full text	Spec. Pur.	Yes	No	No

[1] Can be user programmed.
[2] Planned for next version.

In this table, each column represents one possible feature or ability that a hypertext system can provide. The negative or affirmative entries in the table indicate whether the corresponding hypertext system meets the standard criteria for a specified feature. These criteria are listed below.

Hierarchy: Is there specific support for hierarchical structures?
Graph-based: Does the system support nonhierarchical (cross-reference) links?
Link types: Can links have types?
Attributes: Can user-designated attribute/value pairs be associated with nodes or links?
Paths: Can many links be strung together into a single persistent object?
Versions: Can nodes or links have more than a single version?
Procedural attachment: Can arbitrary executable procedures be attached to events (such as mousing) at nodes or links?
String search: Can the hyperdocument be searched for strings (including keywords)?
Text editor: What editor is used to create and modify the contents of nodes?
Concurrent multiusers: Can several users edit the hyperdocument at the same time?
Pictures or graphics: Is some form of pictorial or graphical information supported in addition to text?
Graphics browser: Is there a browser which graphically presents the nodes and links in the hyperdocument?

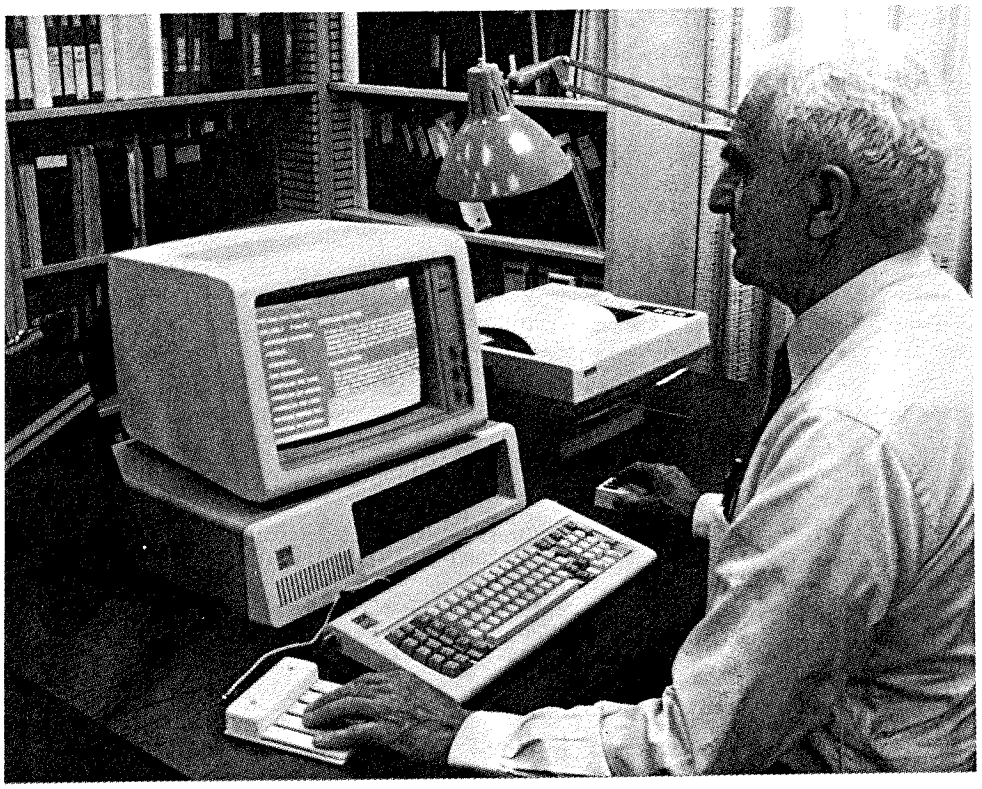


Figure 3. Engelbart at the NLS/Augment workstation. Note the chord key set under Engelbart's left hand. The chord key set is optional for Augment. It is a remarkable accelerator for character-driven commands and mouse-select screen operands.

each viewer; out of view, the code space is also filled with a photocell-readable dot code that names the other document and the current position in that document. Thereafter, when one of these items is in view, the other can be instantly recalled merely by tapping a button below the corresponding code space. Bush admitted that many technological breakthroughs would be needed to make his memex practical, but he felt that it was a technological achievement worthy of major expenditure.

Engelbart's NLS/Augment. Just less than two decades later Douglas Engelbart, at Stanford Research Institute, was influenced by Bush's ideas. In 1963, Engelbart wrote "A Conceptual Framework for the Augmentation of Man's Intellect."[3] Engelbart envisioned that computers would usher in a new stage of human evolution, characterized by "automated external symbol manipulation":

> In this stage, the symbols with which the human represents the concepts he is manipulating can be arranged before his eyes, moved, stored, recalled, operated upon according to extremely complex rules—all in very rapid response to a minimum amount of information supplied by the human, by means of special cooperative technological devices. In the limit of what we might now imagine, this could be a computer, with which individuals could communicate rapidly and easily, coupled to a three-dimensional color display with which extremely sophisticated images could be constructed . . .[3]

His proposed system, H-LAM/T (Human using Language, Artifacts, and Methodology, in which he is Trained), included the human user as an essential element: The user and the computer were dynamically changing components in a symbiosis which had the effect of "amplifying" the native intelligence of the user. This is still a common vision among developers of hypertext systems.

Five years later, in 1968, Engelbart's ideas about augmentation had become more specific, and had been implemented as NLS (oN Line System) by the Augmented Human Intellect Research Center at SRI. NLS was designed as an experimental tool on which the research group developed a system that would be adequate to all of their work needs, by

> placing in computer store all of our specifications, plans, designs, programs, documentation, reports, memos, bibliography and reference notes, etc., and doing all of our scratch work, planning, designing, debugging, etc., and a good deal of our intercommunication, via the consoles.[4]

These consoles were very sophisticated by the standards of the day and included television images and a variety of input devices, including one of Engelbart's best known inventions, the mouse.*

Files in NLS were structured into a hierarchy of segments** called *statements*, each of which bore an identifier of its level within the file. For example, a document might have statements "1," "1a," "1a1," "1a2," "1b," etc., though these identifiers did not need to be displayed. Any number of reference links could be established between statements within files and between files. Note that this is a structure which is primarily hierarchical, but which allows nonhierarchical links as well. The importance of supporting both kinds of structures is a point to which I will return later. The system provided several ways to traverse the statements in files.

NLS, like other early hypertext systems, emphasized three aspects: a database of nonlinear text, *view filters* which selected information from this database, and *views* which structured the display of this information for the terminal. The availability of workstations with high resolution displays has shifted the emphasis to more graphical depictions of nodes, links, and networks, such as using one window for each node.

NLS provided viewing filters for the file structure: One could clip the level (depth) of hierarchy displayed, truncate the number of items displayed at any level, and write customized filters (in a "high-level content analysis language") that displayed only statements having the specified content. NLS also introduced the concept of

*Engelbart also introduced a five-key handset—a one-handed keyboard. The operator enters alphanumeric text by "chording" the five keys. Although this method is slower than two-handed typing, it has a considerable advantage for short commands when used with a mouse in the other hand.

**Segments were limited to 3000 characters in length.

multiperson distributed conferencing/editing.

NLS has evolved over the years. It is now called Augment (or NLS/Augment) and is marketed as a commercial network system by McDonnell Douglas. In developing NLS, the emphasis has been on creating a consistent environment for "knowledge workers" (that is, office automation for software engineers). The system now includes many forms of computer-supported communication, both asynchronous (email with links to all documents, journaling of ideas and exchanges, bulletin boards, etc.) and synchronous (several terminals sharing the same display, teleconferencing, etc.). It includes facilities for document production and control, organizational and project information management, and software engineering. (See Figures 3 and 4.)

Nelson's Xanadu Project. During Engelbart's development of Augment, another hypertext visionary, Ted Nelson, was developing his own ideas about augmentation, but with an emphasis on creating a unified literary environment on a global scale. Nelson coined the term "hypertext." His thinking and writing are the most extravagant of any of the early workers. He named his hypertext system Xanadu, after the "magic place of literary memory" in Samuel Taylor Coleridge's poem "Kubla Khan." In Xanadu, storage space is saved by the heavy use of links. Only the original document and the changes made to it are saved. The system easily reconstructs previous versions of documents. Nelson describes his objectives as follows:

> Under guiding ideas which are not technical but literary, we are implementing a system for storage and retrieval of linked and windowing text. The *document*, our fundamental unit, can have windows to any other documents. The evolving corpus is continually expandable without fundamental change. New links and windows can continually add new access pathways to old material. Fast proprietary algorithms render the extreme data fragmentation tolerable in the planned back-end service facility.[5]

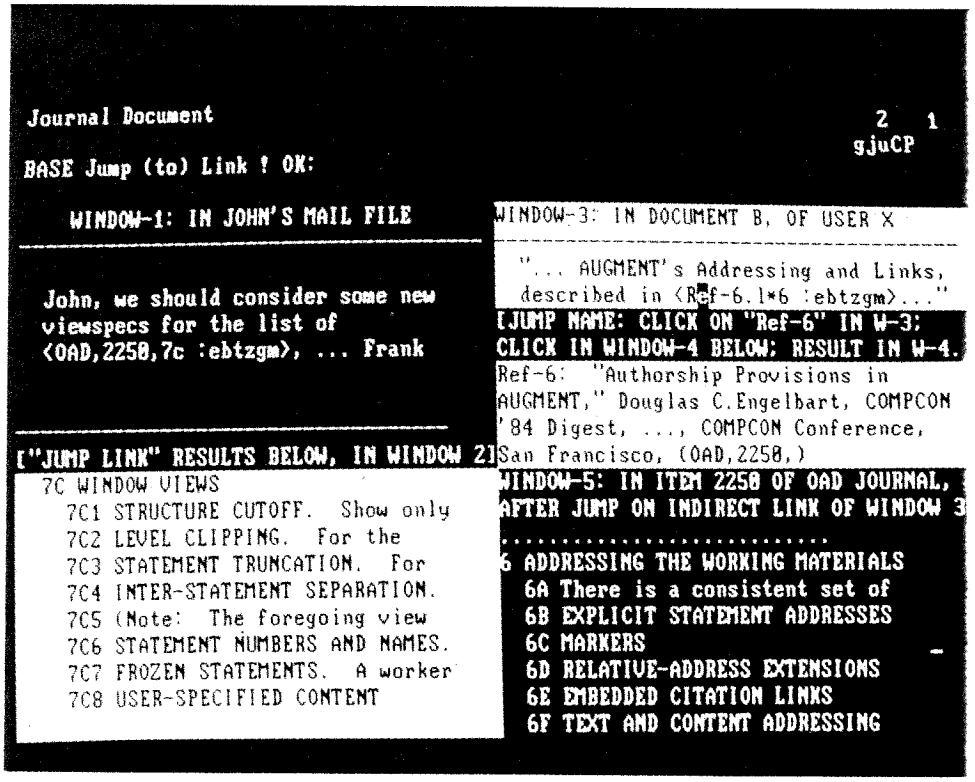


Figure 4. Augment display showing five windows. Window 1 (W-1) has a passage as if embedded in a message, showing a link to Branch 7c of Document 2250 in the OAD Journal. A ViewSpec ("ebtzgm") provides the following specifications: target level plus one, truncate to one line per statement, no blank lines between statements, show only that branch (e.g., not Branch 7d), and turn on Location Numbers. Window 2 (W-2) shows the view obtained with a jump link command. To perform a jump link command, the operator clicks on the link in W-1, then moves the cursor into W-2 for the final click. The very top-left system message announces that the desired Journal Item has been accessed, and the cluster at the top left of the screen verifies that the view is clipped to three levels and the statements truncated to one line each. Window 3 (W-3) shows an indirect link that specifies the linkage path. In effect, this link says "go to the statement in the file named 'Ref-6,' follow the link found there to its target file, and in that file find Location Number 6." Note that the same ViewSpec is specified here as for the link in W-1. Window 4 (W-4) identifies Ref-6 and provides its general reference source as the reference section at the end of the document; a user can jump from the link citation in W-3 to see this statement by using the jump name command. To perform this command, he clicks on "Ref-6" in W-3 then clicks on W-4. Window 5 (W-5) shows a view in the OAD-Journal Item 2250. The user can obtain this view by performing a jump link command on the indirect link of W-3. To perform this command, the user clicks on the indirect link of W-3 and then clicks in W-5.

The long range goal of the Xanadu project has been facilitating the revolutionary process of placing the entire world's literary corpus on line. In fact, Xanadu's design makes a strong separation between the user interface and the database server, with most of the emphasis placed on the latter. In particular, great care has been taken that copyright protection is maintainable, and that a system for the electronic accounting and distribution of royalties is in place. Nelson predicts that the advent of on-line libraries will create a whole new market for the organization and indexing of this immense information store.

The back end of the Xanadu system has been implemented in Unix and is available in several forms, including as an on-line service (much like Engelbart's Augment). A crude front end for the Xanadu system is also available which runs on Sun workstations.

Trigg's Textnet. Randall Trigg wrote the first and to date the only PhD thesis on hypertext. In his thesis, he describes his Textnet system as supporting *nonlinear text*—text in which documents are organized as "primitive pieces of text connected with typed links to form a network similar in many ways to a semantic net." The thesis focuses on specific link types that support literary criticism.

In the tradition of the field, Trigg's system is just a first step in the direction of his vision:

> In our view, the logical and inevitable result [of the computer revolution] will be the transfer of all such [text handling] activities to the computer, transforming communication within the scientific community. All paper writing, critiquing, and refereeing will be performed on line. Rather than having to track down little-known proceedings, journals or unpublished technical reports from distant universities, users will find them stored in one large distributed computerized national paper network. New papers will be written using the network, often collaborated on by multiple authors, and submitted to on-line electronic journals.[6]

Textnet implements two basic types of nodes: those which have textual content (*chunks*) and those which hierarchically organize other nodes (*tocs*, for "table of contents"). Thus Textnet supports both hierarchical trees (via the toc nodes) and nonhierarchical graphs (via the typed links).

Trigg further proposes a specific taxonomy of link types for use by collaborators and critics in Textnet. He argues that there is generally a specific set of types of comments, and that there is a link type for each comment. For example, there are *refutation* and *support* links, and, more specifically, there are links to say that a point is irrelevant ("Pt-irrelevant"), that data cited is inadequate ("D-inadequate"), or that the style is rambling ("S-rambling"). Trigg describes over 80 such link types and argues that the disadvantage of having a limited set of link types is outweighed by the possibility of specialized processing on the hyperdocument afforded by a definite and fixed set of primitives.

In addition, Textnet supports the definition of *paths*—ordered lists of nodes used to browse linear concatenations of text and to dump such scans to hard copy. The path facility relieves the hypertext reader from having to make an *n-way* decision at each link; rather, the reader is provided a default pathway through the network (or part of the network), and can simply read the material in the suggested order as if he were reading a linear document.

Trigg joined Xerox PARC after completing his thesis and was one of the principal architects of the Xerox NoteCards system.

Problem exploration systems. These are highly interactive systems which provide rapid response to a small collection of specialized commands for the manipulation of information. They can be thought of as the early prototypes of electronic spreadsheets for text and symbolic processing. One important feature of most of these tools is a facility for suppressing detail at various levels specified by the user. For example, the outline processors all have single keystroke commands for turning on and off the display of subsections of a section. This is an unusual but natural facility. Hypertext and similar tools excel at the collection of large amounts of relatively unstructured information. But such collections are of little use unless adequate mechanisms exist for filtering, organizing, and browsing. These are the primary desiderata of these authoring/thinking/programming systems.

Issue-Based Information Systems. Horst Rittel and his students have introduced the notion of Issue-Based Information Systems (IBIS)[7] to handle systems analysis in the face of "wicked problems." Rittel describes wicked problems (as opposed to "tame" ones) as problems which cannot be solved by the traditional systems analysis approach (that is, (1) define the problem, (2) collect data, (3) analyze the data, (4) construct a solution). Wicked problems lack a definitive formulation; their problem space cannot be mapped out without understanding the solution elements; in short, the only way to really understand a wicked problem is to solve it. Wicked problems have no stopping rule. The design or planning activity stops for considerations that are external to the problem (for example, lack of time, money, or patience). Solutions to wicked problems are not "right" or "wrong"; they just have degrees of sufficiency. Rittel argues that solving wicked problems requires all those involved to exchange and argue their many viewpoints, ideas, values, and concerns. By coming to understand other viewpoints better, each participant is able to understand the whole problem better. This process enables a common understanding of the major issues and their implications to emerge. IBIS is designed to support this design/planning conversation.

IBIS systems are thus a marriage of (1) teleconferencing systems which enable many people to participate in one conversation, and (2) hypertext, which allows participants to move easily between different issues and the different threads of argument on the same issue. The current version of Rittel's IBIS runs on an Apple PC and is being ported to Sun workstations.* IBIS has three types of nodes (*issues*, *positions*, and *arguments*), and uses nine types of relations to link these nodes. In a typical application, someone posts an issue; then that person or others post positions about that issue; and then the positions are argued using argument nodes. Of course, any of the three types of nodes can be the seed of a new issue. (See Figure 5.) The current set of relationships between nodes is: *responds-to*, *questions*, *supports*, *objects-to*, *specializes*, *generalizes*, *refers-to*, and *replaces*. The research on IBIS concentrates on ways to summarize and present the issue network, both for participants and decision makers.

Lowe's SYNVIEW. David Lowe's SYNVIEW system is similar in concept to Rittel's IBIS but goes in a different direction. It proposes that the participants, in addition to posting their own issues and arguments, assess previous postings as to their validity and relevance. The assessment is done by a kind of quantitative voting. For example, if you think that Joe's response to Sam makes a good point but is not really a direct response to Sam's posting, you might grade it "5, 1" (where 5 is a high validity rating and 1 is a low relevance rating). These values are averaged into the existing values for that posting. The various displays of the argument structure show the values for each posting, allowing readers to focus, if they choose to, on those argument trails having the highest voted validity.

> Through debates on the accuracy of information and on aspects of the structures themselves, a large number of users can cooperatively rank all available items of information in terms of significance and relevance to each topic. Individual users can then choose the depth to which they wish to examine these structures for the purposes at

*A graphical Sun version, called gIBIS, is also being developed at the MCC/Software Technology Program.

hand. The function of this debate is not to arrive at specific conclusions, but rather to collect and order the best available evidence on each topic.[8]

UNC's WE. A group at the University of North Carolina at Chapel Hill has been developing a *writing environment* called WE.[9] Their research is based on a cognitive model of the communication process which explains reading as the process of taking the *linear* stream of text, comprehending it by structuring the concepts hierarchically, and absorbing it into long-term memory as a *network*. Writing is seen as the reverse process: A loosely structured network of internal ideas and external sources is first organized into an appropriate hierarchy (an outline) which is then "encoded" into a linear stream of words, sentences, etc.

WE is designed to support the upstream part of writing. It contains two major view windows, one graphical and one hierarchical, and many specialized commands for moving and structuring material (nodes and links with attached text) between these two views. Normally a writer will begin by creating nodes in the graph view, where he can place them anywhere within the window. At this stage, little or no structure is imposed on the conceptual material. The writer can place nodes in "piles" if they seem to be related, or he can place individual nodes between two piles if they are somewhat related to both. As some conceptual structure begins to emerge from this process, the writer can copy nodes into the hierarchy window, which has specialized commands for tree operations. The hierarchy window has four different display modes: (1) the tree can be laid out on its side, with the root node on the left; (2) the tree can be hung vertically with the root at the top; (3) child nodes can be displayed inside their parent node; and, (4) the hierarchy can be shown in the traditional outline view.

WE uses a relational database for the storage of the nodes and links in the network. The user points with a mouse to select a node. A third window is an editor for the material within the currently selected node. A fourth window on the screen is for queries to the database. A fifth window is used to control system modes and the current working set of nodes.

WE is designed to be an experimental platform to study what tools and facilities will be useful in a writer's environment. The real validation of these ideas, as with so many of the systems described here, will come with further experiments and analysis.

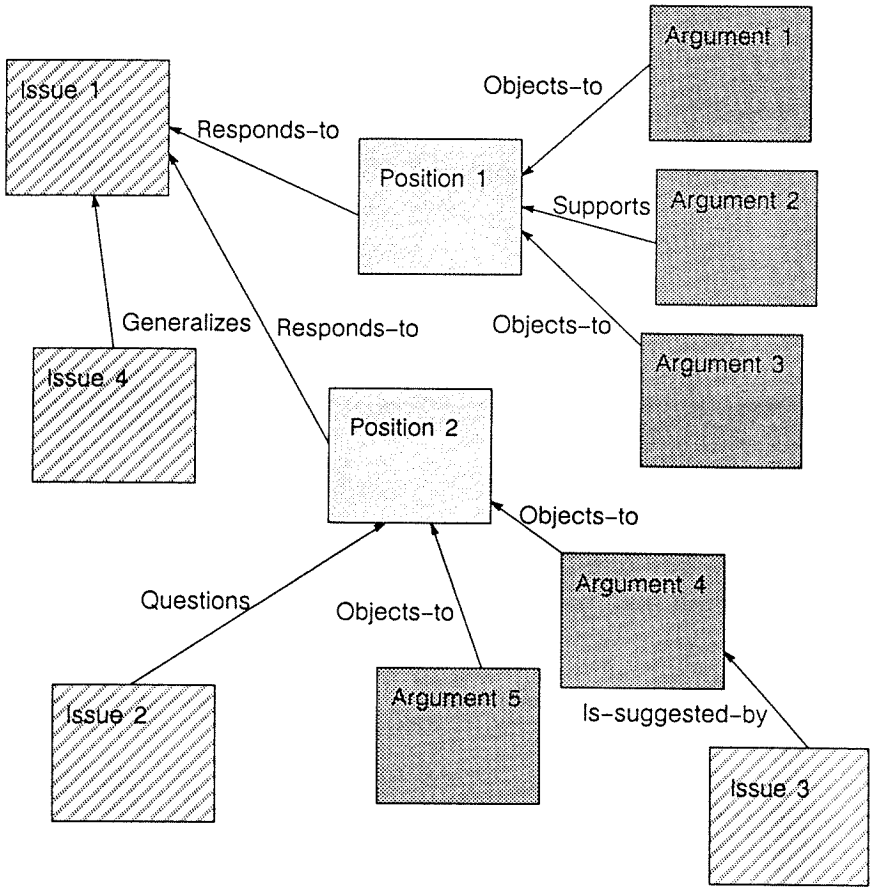


Figure 5. A segment of a possible IBIS-style discussion showing the topology of the IBIS network. Each node contains information on the type of the node, the time and date of creation, the author, a short phrase describing the content, a longer body of text with the text of the comment, a list of keywords, and a list of the incoming and outgoing links.

Outline processors. An *outline processor* is a word processing program which is specialized for processing outlines, in that its main commands deal with movement among, creation of, and modification of outline entries. In this respect, these programs commercialize many ideas from Engelbart's NLS/Augment. Outline processors also have at least simple text editors and do some text formatting, so that the user can use the same tool to go from outline to finished document. One of the most powerful features of outline processing is the ability to suppress lower levels of detail in the outline. As with Engelbart's NLS/Augment, the user can view just the top level of the outline, or the top *n* levels, or he can "walk the tree," opening up just those entries that are relevant or useful to the idea that he is working on. In addition, each outline entry can have a textual body of any length associated with it, and the user can make this body appear or disappear with a single keystroke. This feature is a real boon to the writing process, because it allows the user to have a view of both the immediate text that he is composing and the global context for it. It also facilitates rapid movement between sections, particularly in large documents, because in outline mode a remote section is never more than a few keystrokes away.

Most outline processors are personal computer programs, and they have done much to bring some of the concepts underlying hypertext into popularity. The first of these was called ThinkTank. It was released in 1984. It has since been joined

by a host of others, with names like MaxThink, Executive Writer/Executive Filer, Thor, Framework, Kamas, Fact Cruncher, Freestyle, Idea!, and PC-Outline.[10] There are two very recent additions to the field: Houdini is an extension of MaxThink that supports rich nonhierarchical internode references; and For-Comment is a word processor that allows up to 15 people to apply hypertext-like annotations to a document (and can operate over a Local Area Network (LAN) in real time).

Aside from Houdini, most outline processors do not support inter-entry references, except by "cloning" the whole entry and displaying it in the new location. Only a few others provide windows for nodes. None of them provide explicit "mousable" link icons. For these reasons, one could argue whether they qualify as hypertext as I have defined it here. However, ThinkTank was the first program to be billed—somewhat pretentiously—as an "idea processor," and all of these programs treat sections of text as first-class objects and support manipulations that coincide with the way one manages ideas. They share these features with hypertext, and in this sense, they anticipate the inevitable proliferation of hypertext features within the mainstream of computer applications.

Structured browsing systems. The systems reviewed in this section were designed primarily for applications involving large amounts of existing information or requiring easy access to information. These systems pose different problems for their designers. Ease of learning and ease of use are paramount, and great care goes into crafting the interface. On the other hand, writing (adding new information) is usually either not allowed to the casual user or is not particularly well supported.

CMU's ZOG and Knowledge Systems' KMS. ZOG is a menu-based display system developed in 1972 at Carnegie-Mellon University.[11] It consists of a potentially large database of small (screen-sized) segments which are viewed one at a time. ZOG was developed with the particular goal of serving a large simultaneous user community, and thus was designed to operate on standard terminals on a large timesharing system. In 1981 two of the principals on the ZOG Project, Donald McCracken and Robert Akscyn, started the company Knowledge Systems and developed a commercial successor to ZOG called Knowledge Management System (KMS).

Each segment of the ZOG/KMS database is called a *frame*. A frame has, by convention, a one-line title at the top of the screen, a few lines of text below the title stating the issue or topic of the frame, a set of numbered (or lettered) menu items of text called *selections*, and a line of standard ZOG commands called *global pads* at the bottom of the screen. (Some of these commands are: *edit*, *help*, *back*, *next*, *mark*, *return*, and *comment*.) The selections interconnect the frames. When a user selects an item by typing its number or letter at the terminal keyboard, the selected frame appears on the screen, replacing the previous frame. The structure is generally hierarchical, though cross-referencing links can be included. In addition, an item in a frame can be used to activate a process.

In 1982 ZOG was installed and used as a computer-based information management system on the nuclear-powered aircraft carrier USS CARL VINSON. This system is probably the largest and most thoroughly tested hypertext system in service in the field. ZOG has also been used for more interactive process applications such as policy analysis, authoring, communications, and code management. Historically, however, ZOG made its name more as a bulletin board/textual database/CAI tool than as an interactive system. Hence it is included in this section on browsing. A drawback of the ZOG/KMS style of viewing a single frame at a time is that users may become disoriented, since no spatial event corresponds to the process of moving from frame to frame. In the KMS system, this tendency has been offset by minimizing system response time, so that frame-to-frame transition takes about half a second. The possibility of user disorientation is greatly reduced by the fact that the user can move very quickly among frames and thus become reoriented with very little effort. Creating text and graphics is also fast in KMS.

Emacs INFO Subsystem. The help system in the widely used text editor, Emacs, is called INFO, and is much like ZOG. It has a simpler set of standard commands, and its control input is done by single letters or short commands typed at the keyboard. It is primarily hierarchical, but a user can jump to a different place in the hierarchy by typing in the name of the destination node. It is used as an on-line help system in Emacs. INFO has the same potential for user disorientation which is shared by all of the systems which display only a single frame at a time and have no browser.

Shneiderman's Hyperties. The University of Maryland Hyperties project* has been developed in two directions—as a practical and easy-to-learn tool for browsing in instructional databases and as an experimental platform for studies on the design of hypertext interfaces. As a practical tool, it has already seen some use in the field at a Washington, D.C. museum exhibit about Austria and the Holocaust. (See Figure 6.) Designers of the exhibit emphasized making the system easy and fun for users who have never used a computer before. As an experimental platform, it has been used in five experimental studies involving over 220 subjects.[12]

In Hyperties the basic units are short articles (50-1000 words typically), which are interconnected by any number of links. The links are highlighted words or phrases in the article text. The user activates the links by touching them with a finger (on a touch-sensitive screen) or using the arrow keys to jump to them.** Activating a link causes the article about that topic to appear in its own window on the screen. The system keeps track of the user's path through the network of articles, allowing easy return from exploratory side paths.

In addition to a title and a body of text, each article has a short (5- to 25-word) description which the program can display very quickly. This feature allows the user an intermediate position between bringing up the full article and trying to guess from the link name precisely what the article is about.

Hyperties runs on the IBM PC. Recently graphics capabilities have been added to the system. Current implementation efforts focus on support for videodisc images. Also, a browser is being developed

*The "ties" in "Hyperties" stands for "The Interactive Encyclopedia System."

**The Hyperties system uses a different convention than the mouse to select links. In the Hyperties system, some link is always selected. When the user pushes one of the arrow keys, the system responds by selecting the nearest link in the direction of the arrow. Studies showed this to be a faster and easier technique for selecting arbitrary highlighted fields on the screen.

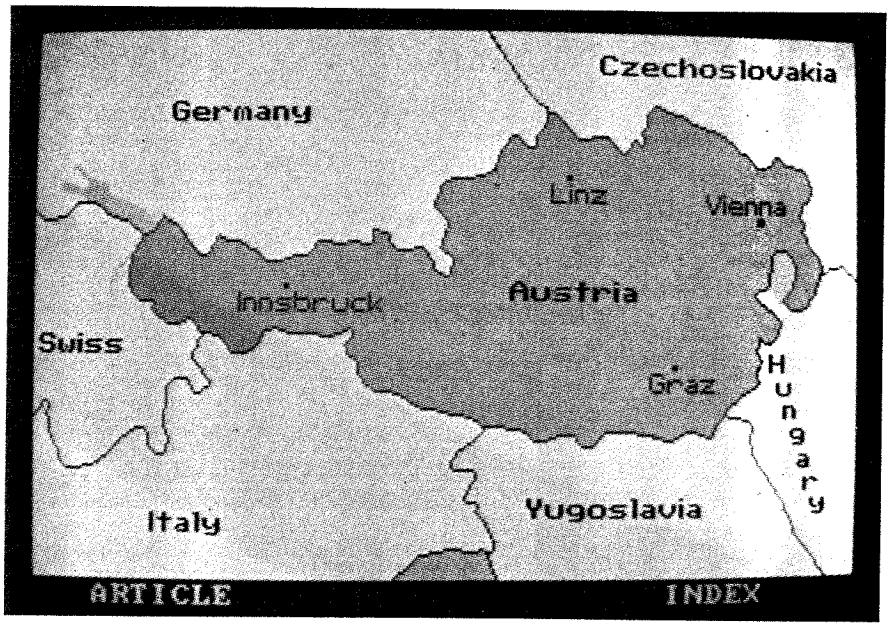
which will provide string search, bookmarks, multiple windows, and user annotation.

Symbolics Document Examiner. The most advanced of the on-line help systems, this tool displays the pages from the entire twelve-volume manual set on the Symbolics Lisp machine screen.[13] Certain textual fields in the document (printed in bold) are mouse-sensitive. Touching one of these fields with the mouse causes the relevant section of the manual to be added to the current working set of manual pages. The system allows the reader to place *bookmarks* on any topic and to move swiftly between bookmarked topics. The protocol for link following is tailored to browsing in a reference manual or encyclopedia. Mousing a link only causes it to be placed on a list of current topics. Then, mousing an entry in this list causes that link to be followed, bringing up the referenced topic in the main viewing window.

The system also supports on-line string search of preidentified keywords, including the search for whole words, leading substrings, and embedded substrings. The system is thus well designed for the specific task of browsing through a technical manual and pursuing several aspects of a technical question or several levels of detail simultaneously. The user cannot make any changes or additions to the manual set (although it is possible to save personalized collections of bookmarks).

General hypertext technology. So far I have discussed hypertext systems that have particular practical applications. The following systems also have one or more applications, but their primary purpose is experimentation with hypertext itself as a technology. For example, while NoteCards has been used for authoring, programming, personal information management, project management, legal research, engineering design, and CAI, its developers view it primarily as a research vehicle for the study of hypertext.

Xerox PARC's NoteCards. Perhaps the best known version of full hypertext is the *NoteCards* system developed at Xerox PARC.[14] The original motivation in building NoteCards was to develop an information analyst's support tool, one that would help gather information about a topic and produce analytic reports. The designers of Notecards observed that an information analyst usually follows a general procedure that consists of a series



PLACES: AUSTRIA PAGE 1 OF 3

Austria (see map) holds a special place in the history of the Holocaust. Situated between Eastern and Western Europe, possessing a vibrant and culturally creative **Jewish community** on the eve of World War II, Austria had also provided the young **Adolf Hitler**, himself an Austrian raised near Linz, with important lessons in the political uses of **antisemitism** Leading Nazis came from Austria: the names of Adolf Hitler, **Adolf Eichmann**, who organized the **deportations** of the Jews to the **death camps**, and **Ernst Kaltenbrunner**, the head of the Reich Main Office for Security, 1943-45, readily come to mind. As

---

Linz - city in northern Austria; childhood home of Adolf Hitler and other leading Nazis

NEXT PAGE **RETURN TO GYPSIES** **INDEX**

Figure 6. The Hyperties Browser enables users to traverse a database of articles and pictures by selecting from highlighted items embedded in the text of the articles. The photos show the IBM PC version of Hyperties. The upper node shows a map of Austria. The lower node shows double-spaced text with link terms highlighted. Either a touchscreen or jump-arrow keys are used for selection of brief definitions, full articles, or pictures. The Hyperties Author permits people with only word processing skills to create and maintain databases. Research versions of Hyperties run on the Enhanced Graphics Adapter to give more lines and multiple windows and on the Sun 3 workstation to show two full pages of text at a time. Current development efforts will enable readers to point at pictures and videodisc images to retrieve further information.

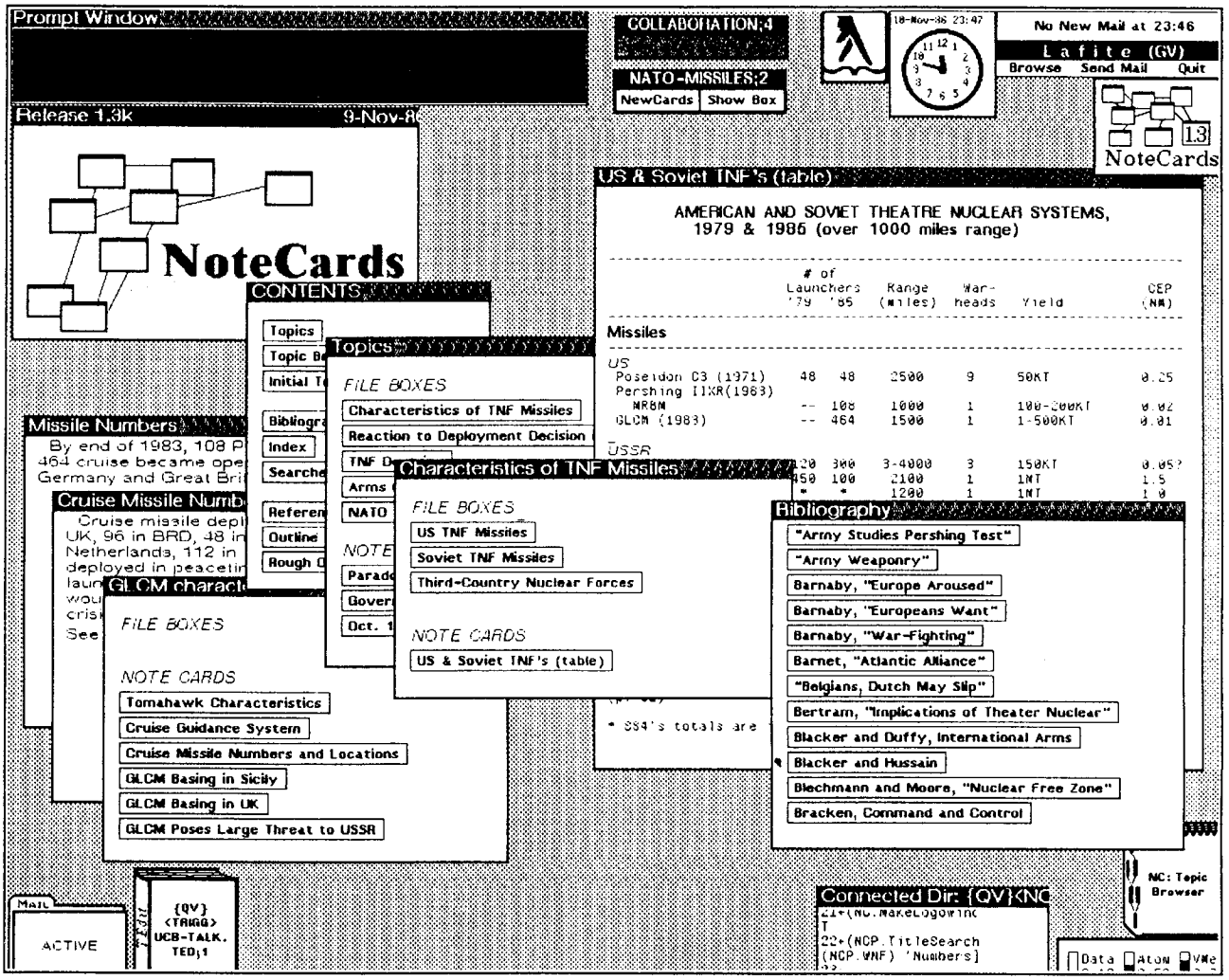


Figure 7. A typical NoteCards screen with five FileBox cards, two unformatted Text cards, and one Text card formatted as a table. Links between cards are represented by the boxed text inside the cards. The two menus at the top/middle of the screen control two different note files. The remainder of the icons on the screen belong to non-NoteCard applications running in the Xerox Lisp environment.

of steps: (1) reading sources (news reports, scholarly articles, etc.), (2) collecting clippings and filing them (in actual shoeboxes!), and (3) writing analytic reports. The designers also observed that throughout the process, the analyst forms analyses and conceptual models in his head. The research goal of the PARC team was to develop technology to aid the analyst in forming better conceptual models and analyses, and to find better expressions of these models and analyses.

A programmer's interface makes NoteCards an open architecture that allows users to build (in Lisp) new applications on top of NoteCards. Using this interface, the user can easily customize the browser. NoteCards allows easy creation of new types of nodes. Forty or fifty such specialized node types have been created to date, including text, video, animation, graphics, and actions.* The new version also allows several users to work in the same Notefile at the same time.

Part of NoteCards' success is due to the fact that it was developed on Xerox D-series Lisp machines, which are powerful workstations that have high resolution screens allowing windows and link and node icons to be displayed in very high resolution. (See Figure 7.) Currently between 50 and 100 users use NoteCards, many of them outside of Xerox (even though it is not a supported product). Several of these users have constructed very large databases in the system (for example, 1600 nodes with 3500 links between them).

Brown University's Intermedia. One of the oldest and largest hypertext research groups exists at Brown University, at the Institute for Research in Information and Scholarship (IRIS).[15] The Intermedia

*An *action node* contains Lisp code which gets evaluated when a link to the node is activated.

project builds on two decades of work and three prior generations of hypertext systems.[16]

The first system was the Hypertext Editing System designed by Ted Nelson, Andy van Dam, and several Brown students for the IBM 2250 display in 1968. This system was used by the Houston Manned Spacecraft Center to produce Apollo documentation.

The second system was the File Retrieval and Editing System (FRESS). FRESS was a greatly enhanced multiterminal timesharing version designed by van Dam and his students. It became available in 1969 and was commercially reimplemented by Phillips in the early 1970's. FRESS was used in production by hundreds of faculty and students over more than a decade. Its users included an English poetry class that did all of its reading and writing on a communal hypertext document. Like NLS, FRESS featured both dynamic hierarchy and bidirectional reference links, and keyworded links and nodes. Unlike NLS, it imposed no limits on the sizes of nodes. On graphics terminals, multiple windows and vector graphics were supported.

The third project, the Electronic Document system, was a hypermedia system emphasizing color raster graphics and navigation aids.

As part of Brown's overall effort to bring graphics-based workstations into effective use within the classroom, the Intermedia system is being developed as a framework for a collection of tools that allow authors to create links to documents of various media such as text, timelines, diagrams and other computer-generated images, video documentaries, and music. Two courses, one on cell biology and one on English literature, have been taught using the system. Current applications include InterText, a text processor; InterDraw, a graphics editor; InterVal, a timeline editor that allows users interactively to organize information in time and date sequences; InterSpec, a viewer for sections of 3D objects; and InterPix, a scanned-image viewer. Under development are a video editor, a 2D animation editor, and more complex methods for filtering the corpus and creating and traversing trails.

Intermedia is being developed both as a tool for professors to organize and present their lesson material via computer and as an interactive medium for students to study the materials and add their own annotations and reports.

For example, in the English literature course the first time a student is searching for background information on Alexander Pope, he or she may be interested in Pope's life and the political events that prompted his satiric criticism. To pursue this line of thought the student might retrieve the biography of Pope and a timeline summarizing political events taking place in England during Pope's life. Subsequently, the student may want to compare Pope's use of satire with other later authors' satiric techniques. This time the student may look at the same information about Pope but juxtapose it with information about other satiricists instead of a time line. The instructor (and other students, if permitted) could read the student's paper, examine the reference material, and add personal annotation links such as comments, criticism, and suggestions for revision. While revising the document, the student could see all of the instructor's comments and examine the sources containing the counter-arguments.

Like most of the serious workers on hypertext, the Intermedia team is especially concerned with providing the user with ways of managing the increased complexity of the hypertext environment. For example, they contend that multiple links emanating from the same point in a document may confuse the reader. Their alternative is to have a single link icon in the material (text or graphics) which can be quickly queried via the mouse to show the specific outgoing links, their names, and their destination nodes.[15] They also propose a construct called a *web* to implement context-dependent link display. Every link belongs to one or more webs and is only visible when one of those webs is active. To view documents with the links that belong to a particular web, a user opens a web and then opens one or more of its documents. Although other webs may also reference the document, only the links which were made in the current web are displayed. As a result, the user does not have to sift through the connections made in many different contexts.

The Intermedia project is also studying ways of providing an effective browser for a network that can include hundreds or even thousands of nodes. The Intermedia browser has two kinds of displays: a *global map*, which shows the entire hyperdocument and allows navigation within it; and a *local map*, which presents a view centered on a single document and displaying its links and nearest neighbors in the web. In addition, a display can show nodes and links at several levels of detail. For example, it can show whole documents and the links between them, or each link and its approximate location within its documents. (See Figure 8.)

The Intermedia project has a long history, many participants, and a serious institutional commitment to long-term objectives. It conducts creative hypertext experiments and uses the classroom as a proving ground. Although this project is still in its early stages, we can expect it to contribute significantly to the development of effective cooperative work environments based on hypertext.

Tektronix Neptune. Tektronix Neptune is one hypertext system that has been particularly designed as an open, layered architecture.[17] Neptune strongly separates the front end, a Smalltalk-based user interface, from the back end, a transaction-based server called the Hypertext Abstract Machine (HAM). The HAM is a generic hypertext model which provides operations for creating, modifying, and accessing nodes and links. It maintains a complete version history of each node in the hyperdocument, and provides rapid access to any version of a hyperdocument. It provides distributed access over a computer network, synchronization for multiuser access, a complex network versioning scheme, and transaction-based crash recovery.

The interface layer provides several browsers: A *graph browser* provides a pictorial view of a subgraph of nodes and links; a *document browser* supports the browsing of hierarchical structures of nodes and links; and a *node browser* accesses an individual node in a hyperdocument. Other browsers include *attribute browsers*, *version browsers*, *node differences browsers*, and *demon browsers*. (See Figure 9.)

In Neptune, each end of a link has an offset within its node, whether that node is textual or graphical. * The link attachment may refer to a particular version of a node, or it may refer to the current version. The HAM provides two mechanisms that are useful for building application layers: Nodes and links may have an unlimited number of attribute/value pairs; and special high-speed predicates are included for querying the values of these pairs in the entire hyperdocument, allow-

*Unlike in most hypertext systems, the destination end of a Neptune link is an iconic point in the text of the destination node rather than the whole node.

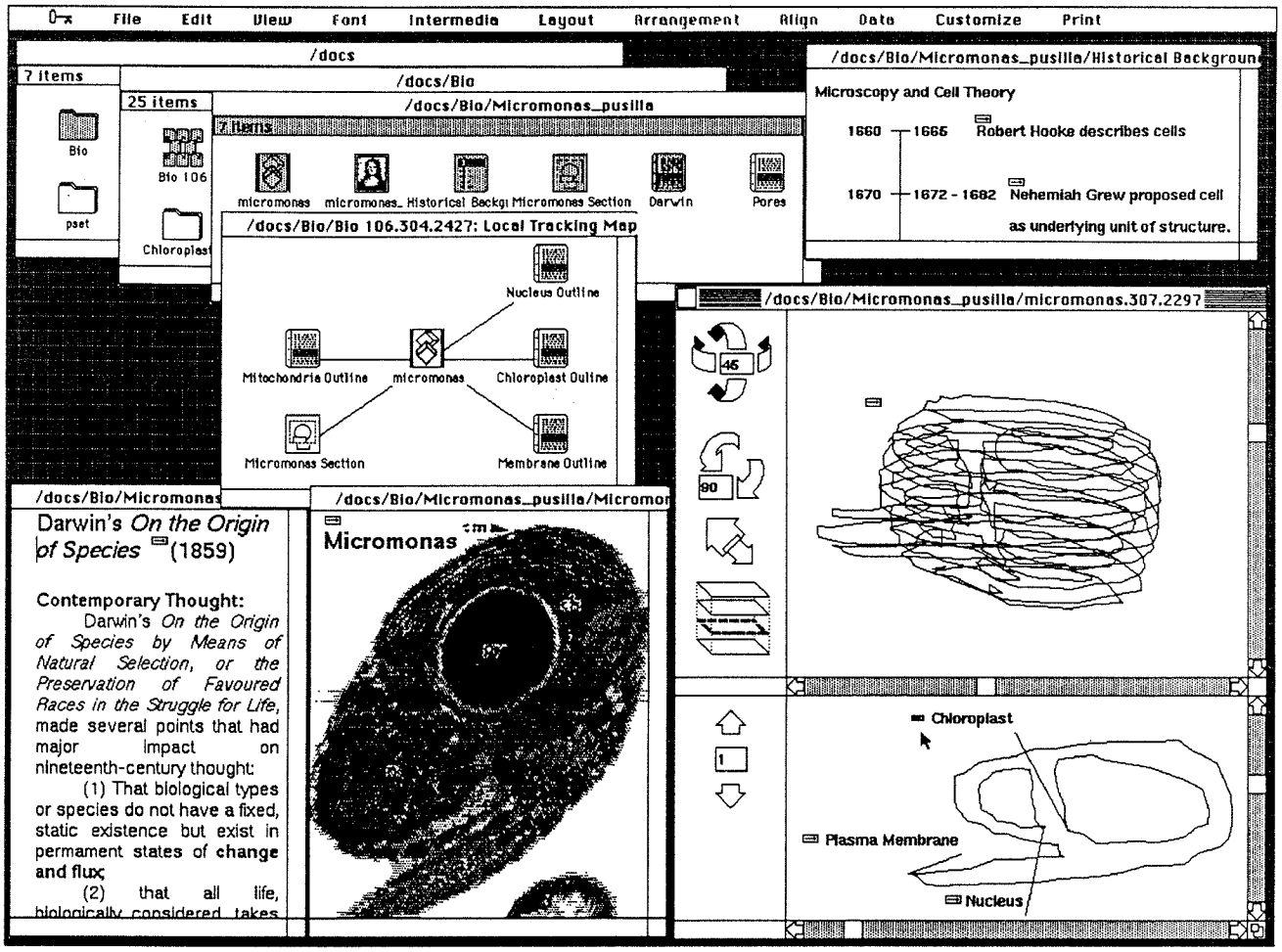


Figure 8. The Intermedia System. This figure illustrates materials from an Intermedia corpus called "Bio 106: Cell Biology in Context." Three folder windows containing hierarchically organized documents of different types are open in the upper left side of the display. An InterText document (lower left side) and an InterVal document (upper right side) are currently open, as well as an InterDraw document containing a scanned electronmicrograph (lower middle). This image has been linked to a corresponding three-dimensional image displayed in an InterSpect document (lower right). The "lower tracking map" (center) shows the links emanating from the current document. Authors or browsers can manipulate the three-dimensional image, edit text and graphics, follow links or create links at any time in this environment. (The electronmicrograph of *Micromonas* was published in the *Journal of Phycology* and is reprinted with the permission of the Editor.)

ing higher level applications to define their own accessing mechanisms on the graph. The HAM also provides a demon mechanism that invokes arbitrary code when a specific HAM event occurs.

diSessa's Boxer. Boxer[18] is a highly interactive programming language specifically tailored to be easy for noncomputer specialists to learn. Boxer uses a *box* to represent a unit of information in the system. In Boxer, one box can contain other boxes, or data such as text or graphics. For example, a program is a box that contains some boxes that provide input and output variables, and other boxes that specify behavior. The system also supports alternate views of some boxes: A box which specifies a graphics routine can also show that graphic display.

Since Boxer is a programming language, it treats cross-reference links in a special way. Rather than using mousable icons as links, Boxer uses a specialized box, called a "port," which gives a direct view into the destination. For example, a port from box A to box B appears within A as a box which shows B. But a port is more than just a view of the destination box, because the destination box can be changed through any of the ports which lead to it, and the changes will be reflected in all of these ports.

Hierarchy is more naturally expressed in Boxer than in many of the other hypertext systems. Boxes are nested within each other two-dimensionally, and are filtered to reduce the level of clutter on the screen. This system of representation has the

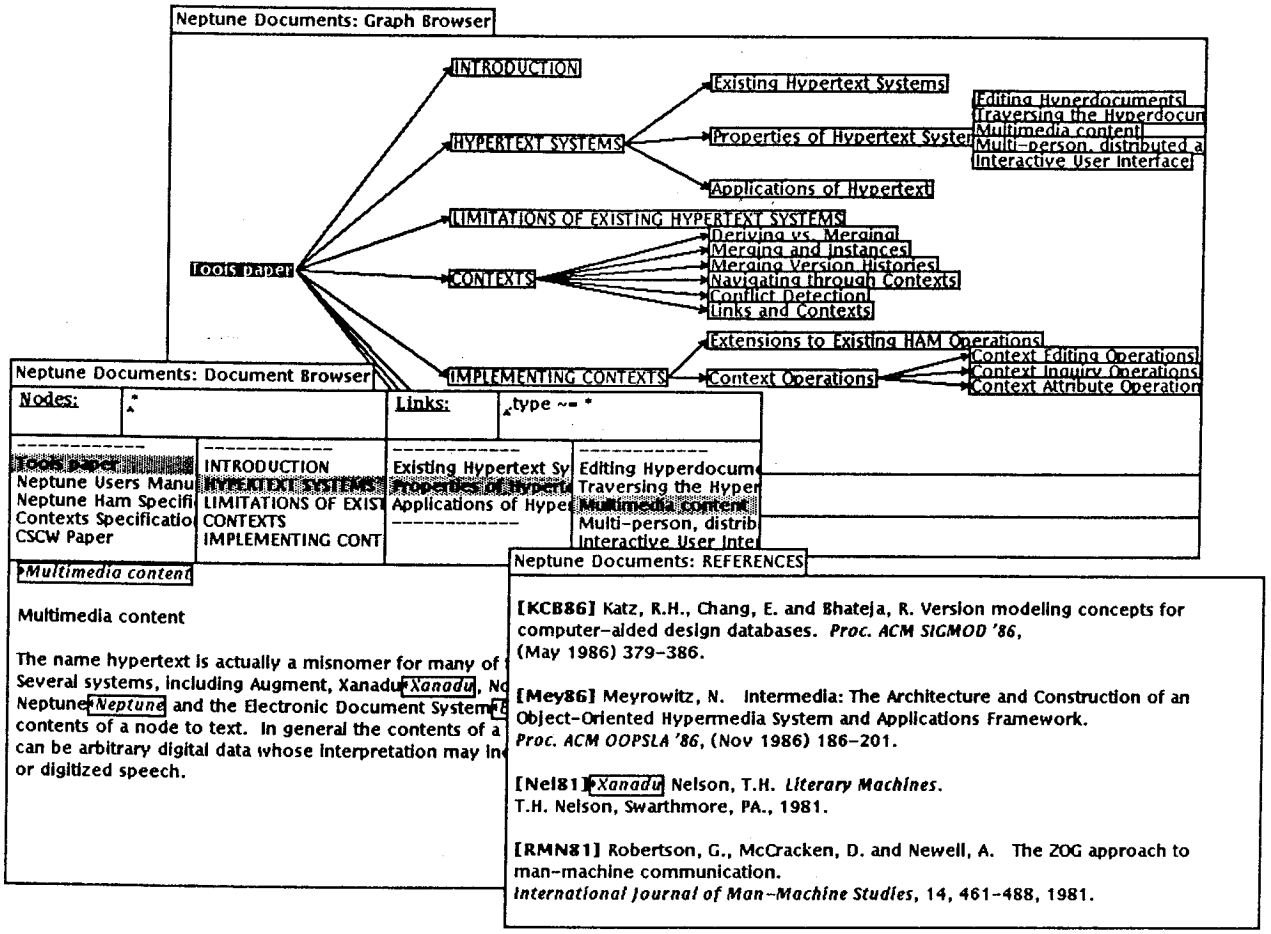


Figure 9. Neptune browsers. Three browsers from Neptune are illustrated. A pictorial view of a network of nodes and links is shown in the Graph Browser (the upper window). The lower right window and the lower pane of the Document Browser are viewers for text nodes. Icons representing link attachments are shown embedded within the text in each of the nodes.

advantage of showing a natural hierarchy of nodes: The windows of lower-level nodes are nested directly within their parents. In most hypertext systems, no attempt is made to display the parent-child relationship once the nodes are opened as windows.

Pitman's CREF. The Cross-Referenced Editing Facility (CREF) is a prototype of a specialized text and graphics editor which was developed originally as a tool for use in analyzing the transcripts from psychological experiments (known as *protocols*), but which was also used to investigate more general hypertext design issues.[19] Much of the interactive feel of CREF reflects the style of use and programming of the Symbolics Lisp machine, on which it was built. Chunks of text, called *segments*, constitute the nodes in the system. Segments are arranged in linear series, and can have keywords and various kinds of links to other segments. The notion of a linear set of segments is natural to the protocol analysis problem, since the first step with such protocols is to segment them into the episodes of the experimental session.

CREF organizes segments into *collections*, which can be defined implicitly by a predicate (called an *abstract collection*) or explicitly by a list (called a *static collection*). At any time, the selected collection appears as a continuous length of text with the segment boundaries marked by named horizontal lines (such as "Segment 1," "Segment 2," etc.). This view can be edited as if it were a single document.

One way of forming an abstract collection is by selecting segments using a boolean predicate over keywords. To extend the power of this keyword facility, CREF allows the user to define a type hierarchy on the keywords. For example, if

"card 105" is defined as a type of (i.e., a *child* of) "card," then collections based on the keyword "card" will also contain segments which have only "card 105" as a keyword.

CREF supports four kinds of links: *references links* cross-reference among segments; *summarizes links* impose hierarchy (a *summary* is a segment which has one or more summarizes links to other segments); *supersedes links* implement versioning by copying the superseded segment and freezing it; and *precedes links* place a linear ordering on segments.

Finally, CREF allows multiple analysts to compose different *theories* about a protocol, using the same segmented data. Each theory imposes its own structure on the data, and has its own collections, diagrams, keywords, and annotations. This mode of selection is similar to the notion of contexts or webs used in other systems.

Hypertext on the Macintosh. At least two programs have been written for the Apple Macintosh that provide hypertext facilities: FileVision and Guide.

FileVision is primarily oriented to graphics nodes and to applications which can exploit visual indexing. The advertising for FileVision describes applications that encourage visual indexing. For example, in the database for a travel agency, the map of a region may contain icons for the main cities in that region. The user clicks on the icon for a city to obtain a display of a map of that city. The map of the city may have icons for the major landmarks in the city. The user clicks on one of these icons to obtain a display of data about the landmark, or perhaps even to obtain a picture of the landmark itself.

Guide is a more recent program which is based on an earlier Unix version developed in England.[20] It does not provide the graphics capabilities of FileVision (graphics are supported but cannot contain links), but it does support textual hypertext data very well. Guide uses three kinds of links: *replacement links*, which cause the text in the current window to be completely replaced by the text pointed to by the link; *note links*, which display the destination text in a pop-up window; and *reference links*, which bring up a new window with the destination text. Guide is now available for PCs as well.

As this article goes to press, there is news that Apple will soon have its own hypertext system, called HyperCards. HyperCards will be similar in some ways to Xerox PARC's NoteCards. It will provide special support for *executable links*, which will give it the flavor of a programming language. HyperCards will be bundled with the system software in new Macintoshes.

MCC's PlaneText. PlaneText, developed in the MCC Software Technology Program (STP), is a very recent addition to the family of general hypertext systems.* PlaneText is based on the Unix file system and the Sun SunView window manager. Each node is a Unix file. Links appear as names in curly brackets ({}) whose display can be turned on and off. Links are implemented as pointers saved in separate files, so that the linked files themselves are not changed by creating hypertext references between them. This design allows for the smooth integration of hypertext into the rest of the Unix-based computational environment, including such tools as Mail and News. It allows for the hypertext annotation of standard source code files. In addition, the Unix file directory system serves as a "free" mechanism for creating hierarchical structures among nodes.**

PlaneText supports color graphics nodes which can be freely linked into a hyperdocument.

Summary. The systems in this section were presented in terms of four broad categories: macro literary systems, problem exploration systems, structured browsing systems, and general hypertext technology. Table 1 summarizes this discussion and provides a breakdown of the various features which current hypertext systems can include.

One additional area of research currently is the development of systems which aid the entire process of design, particularly the informal upstream aspects. Such systems require the features of hypertext problem exploration and structured browsing systems as well as the advanced features of the experimental hypertext technologies. Indeed, this area of investigation may become an important fifth category for hypertext systems of the future.

The history of hypertext presented here suggests that the concept and the advantages of hypertext were clear several decades ago, but that widespread interest in hypertext was delayed until the supporting technology was cheap and readily available. This suggestion may be misleading. Many of the "elders" of the field feel that something else has changed as well. They feel that today computer users easily accept the role of the computer as a tool for processing ideas, words, and symbols (in addition to numbers and mere data), and as a vehicle of interhuman communication. Those theorists who gave presentations of their hypertext systems 20 years ago, using expensive state of the art hardware, report that the computer science community showed little interest. This lack of interest seemed to stem as much from a lack of understanding of the basic concepts of hypertext as from a lack of hardware resources.

If this is so, then the recent upsurge in interest in hypertext may signal that the computer community is now ready to consider its technology as much a tool for communication and augmenting the human intellect as for analysis and information processing. Hypertext is certainly a large step in that direction.

The Essence of Hypertext

It is tempting to describe the essence of hypertext as its ability to perform high-speed, branching transactions on textual chunks. But this is a little like describing the essence of a great meal by listing its ingredients. Perhaps a better description would focus on hypertext as a computer-based medium for thinking and communication.

The thinking process does not build new ideas one at a time, starting with nothing and turning out each idea as a finished pearl. Thinking seems rather to proceed on several fronts at one, developing and rejecting ideas at different levels and on different points in parallel, each idea depending on and contributing to the others.

The recording and communication of such entwined lines of thought is challenging because communication is in practice

*It is perhaps too early to say, however, how PlaneText will rank in the world of hypertext, since it will only be publicly available from the participant companies in the MCC/Software Technology Program. For more information call Bill Stotesbery at MCC, (512) 343-0978.

**The use of an existing tree-structuring mechanism limits any hypertext system to only being able to handle a single hierarchical structure. Single hierarchical organizations may be too limited for some advanced applications.

a serial process and is, in any case, limited by the bandwidth of human linguistic processing. Spoken communication of parallel themes must mark items with stresses, pauses, and intonations which the listener must remember as the speaker develops other lines of argument. Graphical forms can use lists, figures, and tables to present ideas in a less than strictly linear form. These visual props allow the reader/viewer to monitor the items which he must understand together. One of the challenges of good writing, especially good technical writing, is to present several parallel lines of a story or an argument in a way that weaves them together coherently.

Traditional flat text binds us to writing and reading paragraphs in a mostly linear succession. There are tricks for signalling branching in the flow of thought when necessary: Parenthetical comments, footnotes, intersectional references (such as "see Chapter 4"), bibliographic references, and sidebars all allow the author to say "here is a related thought, in case you are interested." There are also many rhetorical devices for indicating that ideas belong together as a set but are being presented in linear sequence. But these are rough tools at best, and often do not provide the degree of precision or the speed and convenience of access that we would like.

Hypertext allows and even encourages the writer to make such references, and allows the readers to make their own decisions about which links to follow and in what order. In this sense, hypertext eases the restrictions on the thinker and writer. It does not force a strict decision about whether any given idea is either within the flow of a paper's stream of thought or outside of it. Hypertext also allows annotations on a text to be saved separately from the reference document, yet still be tightly bound to the referent. In this sense, the "linked-ness" of hypertext provides much of its power: It is the machine processible links which extend the text beyond the single dimension of linear flow.

At the same time, some applications demonstrate that the "node-ness" of hypertext is also very powerful. Particularly when hypertext is used as a thinking, writing, or design tool, a natural correspondence can emerge between the objects in the world and the nodes in the hypertext database. By taking advantage of this object-oriented aspect, a hypertext user can build flexible networks which model his problem (or solution). In this application the links are less important than the nodes. The links form the "glue" that holds the nodes together, but the emphasis is on the contents of the nodes.

From a computer science viewpoint, the essence of hypertext is precisely that it is a hybrid that cuts across traditional boundaries. Hypertext is a *database method*, providing a novel way of directly accessing data. This method is quite different from the traditional use of queries. At the same time, hypertext is a *representation scheme*, a kind of semantic network which mixes informal textual material with more formal and mechanized operations and processes. Finally, hypertext is an *interface modality* that features "control buttons" (link icons) which can be arbitrarily embedded within the content material by the user. These are not separate applications of hypertext: They are metaphors for a functionality that is an essential union of all three.

The power of linking. In the next two sections of this article, I will explore links and nodes in more detail as the basic building blocks of hypertext.

Link following. The most distinguishing characteristic of hypertext is its machine support for the tracing of references. But what qualifies a particular reference-tracing device as a link? How much effort is permissible on the part of a user who is attempting to trace a reference? The accepted lower limit of referencing support can be specified as follows: To qualify as hypertext, a system should require no more than a couple of keystrokes (or mouse movements) from the user to follow a single link. In other words, the interface must provide links which act like "magic buttons" to transport the user quickly and easily to a new place in the hyperdocument.

Another essential characteristic of hypertext is the speed with which the system responds to referencing requests. Only the briefest delay should occur (one or two seconds at most). Much design work goes into this feature in most systems. One reason for this concern is that the reader often does not know if he wants to pursue a link reference until he has had a cursory look at the referenced node. If making this judgement takes too long, the user may become frustrated and not bother with the hypertext links.

However, not all link traversals can be instantaneous. Perhaps as important as rapid response is providing cues to the user about the possible delay that a given query or traversal might entail. For example, some visual feature of the link icon could indicate whether the destination node is in memory, on the disk, somewhere else on the network, or archived off line.

Properties of links. Links can be used for several functions. These include the following:

- They can connect a document reference to the document itself.
- They can connect a comment or annotation to the text about which it is written.
- They can provide organizational information (for instance, establish the relationship between two pieces of text or between a table of contents entry and its section).
- They can connect two successive pieces of text, or a piece of text and all of its immediate successors.
- They can connect entries in a table or figure to longer descriptions, or to other tables or figures.

Links can have names and types. They can have a rich set of properties. Some systems allow the display of links to be turned on and off (that is, removed from the display so that the document appears as ordinary text).

The introduction of links into a text system means that an additional set of mechanisms must be added for creating new links, deleting links,* changing link names or attributes, listing links, etc.

Referential links. There are two methods for explicitly linking two points in hypertext—by reference and by organization. The reference method is a nonhierarchical method. It uses referential links that connect points or regions in the text.

Referential links are the kind of link that most clearly distinguishes hypertext. They generally have two ends, and are usually directed, although most systems support "backward" movement along the link. The origination of the link is called the "link source," and usually acts as the *reference*. The source can logically be either a single point or a region of text. At the other end, the "destination" of the link usually functions as the *referent*, and can

*Link deletion is problematical. For example, what should the policy be for nodes which are stranded when all their links have been deleted? Should they be placed in "node limbo" until the user decides what to do with them?

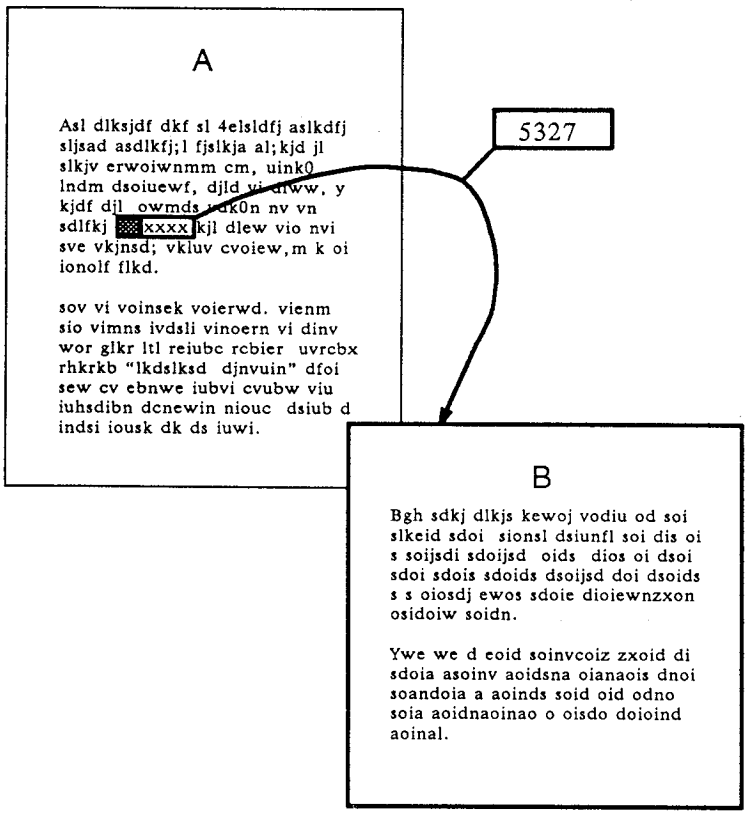


Figure 10. An example of a link with a point source and a region destination. The source of the link is a token in the text of document A which contains a textual identifier ("xxxx"). The identifier may be (1) the name of the destination node (in this case it would be "B"), (2) the name of the link, or (3) an arbitrary string which is neither the name of the link nor the destination node. The destination of this link is node B which is a region. The link has an internal name (5327) which is normally visible to the user.

also be either a point or a region. (See Figure 10.)

A *link point* is some icon indicating the presence of the link. It usually shows the link's name and perhaps also its type. Or it may show the name and/or type of the destination node. In systems such as Neptune which support links with both point source and point destination, the icon also indicates which type of link is indicated. In some systems, the display of links can be suppressed, so that the documents appear linear.

A *link region* is a set of contiguous characters which is displayed as a single unit. In Figure 10, the link destination is a link region, namely, an entire node. Figure 10 illustrates the most common form of hypertext link, in which the source is a point and the destination is a region. This example typifies many of the link applications listed above, because it shows how a chunk of text—a region—is written about or referenced by some smaller chunk of text, often a sentence. Since most readers are accustomed to single point references to sentences (i.e., footnotes), they have no problem accepting a link with a point source. There can be regions in graphics as well—either bordered regions or collections of graphic objects in a figure.

Link regions can pose difficult design problems. They are easiest to implement as whole nodes, since setting a region off from its neighboring material within the same node raises a tough implementation issue—how to display the selected region to the user. It must be highlighted somehow, using reverse video, fonts, or color, but each of these options poses difficulties in keeping overlapping regions clearly highlighted. The Intermedia designers propose to draw a light box around regions and a darker box around region/region overlaps, thus showing a single level of overlapping[15]; however, this technique is not effective if there are more than two overlapping regions.

Another difficulty posed by link regions is how to show the name of the link. Unlike a link point, a link region has no obvious position for a title, unless it is placed arbitrarily at the beginning or end of the region.

Link regions can also be difficult to manipulate. Designers must devise a system for copying, moving, modifying, and deleting the region and the substrings within it. The movement of regions involves logistical dilemmas which are not easy to resolve: For example, when one moves a major portion of the text in a destination region to someplace else in the node, should the link destination move with it or stay with what remains? Also, designers must make special provisions for deleting, moving, or copying the defining end points of a region.

Organizational links. Like reference links, organizational links establish explicit links between points in hypertext. Organizational links differ from referential links in that they implement hierarchical information.

Organizational links connect a parent node with its children and thus form a strict tree subgraph within the hypertext network graph. They correspond to the *IS-A* (or *superconcept*) links of semantic net theory, and thus operate quite differently than referential links.* For example, rather than appearing as explicit highlighted tokens in each node, organizational links are often traversed by a separate mechanism at the node control level (i.e., special *goto-parent*, *goto-first-child*, and *goto-next-sibling* commands). In other cases, there are organizational nodes (such as toc nodes in Textnet and FileBoxes in NoteCards) which record the organizational structure.

*Note that organizational links are distinct from the class hierarchy links that would be used (in the object-oriented programming paradigm) to define types and subtypes of nodes in the hypertext system.

Keyword links. In addition to the explicit linking performed by referential and organizational links, there is a kind of implicit linking that occurs through the use of keywords. This type of linking is yet to be fully explored.

One of the chief advantages of text storage on a computer is the ability to search large and complex documents and sets of documents for substrings and keywords.* Naturally, this ability is also a valuable aspect of hypertext. Indeed, most users of large hyperdocuments insist on having some mechanism for scanning their content, either for selected keywords (which can apply to nodes, links, or regions) or for arbitrary embedded strings.

From a functional standpoint, link following and search are similar: Each is a way to access destination nodes that are of possible interest. Link following usually yields a single node, whereas search can yield many; hence, a keyword is a kind of implicit computed link. The value of this insight is that it may allow design of a hypertext interface which is consistent across all link-tracing activities.

To tree or not to tree. Some hypertext systems (for example, Emacs INFO) support only hierarchical structures, others (such as Xanadu and Hyperties) provide no specific support for hierarchical structures, and others (such as Textnet and NoteCards) support both kinds of structures.

One could question just how sufficient strictly hierarchical structures are, and for which applications they are sufficient and for which they are not. On the one hand, abstraction is a fundamental cognitive process, and hierarchical structures are the most natural structures for organizing levels of abstraction. On the other hand, cases obviously exist where cross-hierarchical links are required. Frank Halasz, one of the developers of NoteCards, has gathered statistics on the *hyperspace* of a single representative NoteCards user; this person had 1577 nodes (cards) in all, 502 of which were FileBoxes (hierarchical nodes). Connecting these nodes were a total of 3460 links, 2521 (73 percent) of which connected FileBoxes to each other or to individual notecards, 261 (7.5 percent) of which were nonhierarchical referential links, and the remainder of which were mail links (used by the system to tie mail messages to other nodes). This example, for what it is worth, suggests that hierarchical structure is very important in organizing a hypertext network, and that referential links are important but less common.

One advantage of a strictly tree-oriented system is that the command language for navigation is very simple: From any node, the most one can do is go to the parent, a sibling, or a child. This simplicity also diminishes the disorientation problem, since a simpler cognitive model of the information space will suffice.

Of course, the great disadvantage of any hierarchy is that its structure is a function of the few specific criteria that were used in creating it. For example, if one wishes to investigate what sea-based life forms have in common with land-based life forms, one may find that the traditional classification of life forms into the plant and animal kingdoms breaks up the information in the wrong way. The creator of a hierarchical organization must anticipate the most important criteria for later access to the information. One solution to this dilemma is to allow the information elements to be structured into multiple hierarchies, thus allowing the world to be "sliced up" into several orthogonal decompositions. Any hypertext system which has hierarchy nodes, such as Textnet (toc nodes) and NoteCards (FileBox nodes), can perform this operation quite easily. These are the only systems which explicitly claim to support multiple hierarchies. Indeed, one early user of NoteCards used the system in doing the research and writing for a major project paper; he imposed one organization on the data and his writings while doing the research, and then quite a different (yet coexistent) organization on the same material to produce his paper. As a generalization, it seems that engineering-oriented hypertext users prefer hierarchical organizations, whereas arts- or humanities-oriented users prefer cross-referencing organizations.

Extensions to basic links. Certain features of the link enable it to be extended in several ways. Links can connect more than two nodes to form *cluster links*. Such cluster links can be useful for referring to several annotations with a single link, and for providing specialized organizational structures among nodes. Indeed, the toc nodes of Textnet and the FileBoxes of NoteCards are both forms of cluster links.

One useful way to extend the basic link is to place attribute/value pairs on links and to query the network for them. The Neptune system, for example, has an architecture that is optimized for this function. Coupled with specialized routines in the database interpreter (the HAM), these attribute lists allow users to customize links in several ways, including devising their own type system for links and performing high-speed queries on the types.

It is also possible to perform procedural attachments on a link so that traversing the link also performs some user-specified side effect, such as customizing the appearance of the destination node. This ability is provided in Neptune and Boxer.

Hypertext nodes. Although the essence of hypertext is its machine-supported linking, the nodes contribute significantly to defining the operations that a hypertext system can perform. Most users of hypertext favor using nodes which express a single concept or idea, and are thus much smaller than traditional files. When nodes are used in this fashion, hypertext introduces an intermediate level of machine support between characters and files, a level which has the vaguely semantic aspect of being oriented to the expression of ideas. But this sizing is completely at the discretion of the hypertext writer, and the process of determining how to modularize a document into nodes is an art, because its impact on the reader is not well understood.[21]

The modularization of ideas. Hypertext invites the writer to modularize ideas into units in a way that allows (1) an individual idea to be referenced elsewhere, and (2) alternative successors of a unit to be offered to the reader (for instance, more detail, an example, bibliographic references, or the logical successor). But the writer must also reckon with the fact that a hypertext node, unlike a textual paragraph, tends to be a strict unit which does not blend seamlessly with its neighbors. Some hypertext systems (Notecards, CREF, Boxer, FRES, NLS) allow nodes to be viewed together as if they were one big node, and this option is essential for some

*There is some controversy over the relative merits of keyword retrieval as opposed to full text search. On the one hand, keyword retrieval is only as good as the skill and thoroughness of the person selecting the keywords. On the other hand, full text search does not find all the relevant documents, nor does it always find only the relevant documents. Its shortcomings are due in part to the commonness of synonyms in English. In addition, full text search can be computationally prohibitive in large networks.

applications (for example, writing and reading prose). But the boundaries around nodes are always discrete and require sometimes difficult judgements about how to cleave the subject matter into suitable chunks.

The process of identifying a semantically based unit, such as an idea or concept, with a syntactic unit, such as a paragraph or hypertext node, is not unique to hypertext. Manuals of style notwithstanding, traditional text has rather loose conventions for modularizing text into paragraphs. This looseness is acceptable because paragraph boundaries have a relatively minor effect on the flow of the reading. Paragraph boundaries are sometimes provided just to break up the text and give the eye a reference point. Thus, decisions about the distribution of sentences among paragraphs is not always critical.

Hypertext, on the other hand, can enforce a rather stern information hiding. In some systems, the only clue a user has as to the contents of a destination node is the name of the link (or the name of the node, if that is provided instead). The writer is no longer making all the decisions about the flow of the text. The reader can and must constantly decide which links to pursue. In this sense, hypertext imposes on both the writer and the reader the need for more process awareness, since either one has the option of *branching* in the flow of the text. Thus hypertext is best suited for applications which require these kinds of judgements anyway, and hypertext merely offers a way to act directly on these judgements and see the results quickly and graphically.

Ideas as objects. While difficult to document, there is something very compelling about reifying the expression of ideas into discrete objects to be linked, moved, and changed as independent entities. Alan Kay and Adele Goldberg[22] observed of Smalltalk that it is able to give objects a perceptual dimension by allocating to them a rectangular piece of screen real estate. This feature offers enhanced retrieval and recognition over computer-processed flat documents, because to a much greater degree abstract objects are directly associated with perceptual objects—the windows and icons on the screen.

Paragraphs, sections, and chapters in a book, viewed through a standard text editor or word processor, don't stand out as first-class entities. This is particularly apparent when one can view one's document hierarchically (i.e., as an outline) at the same time that one adds new sections and embellishes existing ones. People don't think in terms of "screenfulls"; they think in terms of ideas, facts, and evidence. Hypertext, via the notion of nodes as individual expressions of ideas, provides a vehicle which respects this way of thinking and working.

Typed nodes. Some hypertext systems sort nodes into different types. These *typed nodes* can be extremely useful, particularly if one is considering giving them some internal structure, since the types can be used to differentiate the various structural forms.

For example, in our research in the MCC Software Technology Program, we have been implementing a hypertext interface for a design environment called the Design Journal. The Design Journal is intended to provide an active scratchpad in which the designer can deliberate about design decisions and rationale, both individually and in on-line design meetings, and in which he can integrate the design itself with this less formal kind of information. For this purpose we have provided a set of four typed nodes for the designer to use—*notes*, *goals/constraints*, *artifacts*, and *decisions*. Notes are used for everything from reminders, such as "Ask Bill for advice on Module X," to specific problems and ideas relating to the design. Goals/constraints are for the initial requirements as well as discovered constraints within the design. Artifacts are for the elements of the output: The Design. And decisions are for capturing the branch points in the design process, the alternatives considered by the designer, and some of the rationale for any commitment (however tentative) that has been made. The designer captures assumptions in the form of decisions with only one alternative. Our prototype of the Design Journal uses color to distinguish between note types in the browser, and we have found this to be a very effective interface.

Hypertext systems that use typed nodes generally provide a specialized color, size, or iconic form for each node type. The distinguishing features help the user differentiate at a glance the broad classes of typed nodes that he is working with. Systems such as NoteCards, Intermedia, and IBIS make extensive use of typed nodes.

Semistructured Nodes. So far I have spoken of the hypertext node as a structureless "blank slate" into which one might put a word or a whole document. For some applications, there is growing interest in *semistructured nodes*—typed nodes which contain labelled fields and spaces for field values. The purpose of providing a template for node contents is to assist the user in being complete and to assist the computer in processing the nodes. The less that the content of a node is undifferentiated natural language (for example, English) text, the more likely that the computer can do some kinds of limited processing and inference on the textual subchunks. This notion is closely related to Malone's notion of semistructured information systems.[23]

To continue with the example of the Design Journal, we have developed a model for the internal structure of decisions. The model is named ISAAC. It assumes that there are four major components to a design decision:

(1) an *issue*, including a short name for the issue and a short paragraph describing it in general terms;
(2) a set of *alternatives*, each of which resolves the issue in a different way, each having a name and short description, and each potentially linked to the design documents or elements that implement the alternative;
(3) an *analysis* of the competing alternatives, including the specific criteria being used to evaluate them, the trade-off analysis among these alternatives, and links to any data that the analysis draws upon; and
(4) a *commitment* to one of the alternatives (however tentatively) or to a vector of preferences over the alternatives, and a subjective rating about the correctness or confidence of this commitment.

Without getting into the details of the underlying theory, I merely wish to stress here that the internal structure of ISAAC suggests that the author of an ISAAC decision is engaged in a much more structured activity than just "writing down the decision," and the reader is likewise guided by the regularity of the ISAAC structure.

Of course, it may not be clear why we do not treat each of the elements listed above as its own typed hypertext node. The reason is that the parts of an ISAAC frame are much more tightly bound together than ISAAC frames are bound to each other. For example, we could not have an

analysis part without an alternatives part; yet if we treat them as separate hypertext nodes, we have failed to build this constraint into the structure. The general issue here is that some information elements must always occur together, while others may occur together or not, depending on how related they are in a given context and how important it is to present them as a cluster distinct from "surrounding" information elements. This problem is recursive: An element that is atomic at one level may turn out on closer inspection to contain many components, some of which are clustered together.

In hypertext this tension presents itself as the twin notions of semistructured nodes and composite nodes.

Composite nodes. Another mechanism for aggregating related information in hypertext is the *composite node*. Several related hypertext nodes are "glued" together and the collection is treated as a single node, with its own name, types, versions, etc. Composite nodes are most useful for situations in which the separate items in a bulleted list or the entries in a table are distinct nodes but also cohere into a higher level structure (such as the list or table). This practice can, however, undermine the fundamental association of one interface object (window) per database object (node), and thus must be managed well to avoid complicating the hypertext idiom unduly.

A composite node facility allows a group of nodes to be treated as a single node. The composite node can be moved and resized, and closes up to a suitable icon reflecting its contents. The subnodes are separable and rearrangeable through a subedit mode. The most flexible means of displaying a composite node is to use a constraint language (such as that developed by Symbolics for Constraint Frames) which describes the subnodes as *panes* in the composite node window and specifies the interpane relationships as dynamic constraints on size and configuration.

Composite nodes can be an effective means of managing the problem of having a large number of named objects in one's environment. Pitman described the problem this way:

> In this sort of system, there is a never-ending tension between trying to name everything (in which case, the number of named things can grow quickly and the set can become quickly unmanageable) or to name as little as possible (in which case, things that took a lot of trouble to construct can be hard to retrieve if one accidentally drops the pointers to them).[19]

One problem with composite nodes is that as the member nodes grow and change the aggregation can become misleading or incorrect. A user who encounters this problem is in the same predicament as a writer who has rewritten a section of a paper so thoroughly that the section title is no longer accurate. This "semantic drift" can be difficult to catch.

Analogy to semantic networks. The idea of building a directed graph of informal textual elements is similar to the AI concept of *semantic networks*. A semantic network is a knowledge representation scheme consisting of a directed graph in which concepts are represented as nodes, and the relationships between concepts are represented as the links between them. What distinguishes a semantic network as an AI representation scheme is that concepts in the representation are indexed by their semantic content rather than by some arbitrary (for example, alphabetical) ordering. One benefit of semantic networks is that they are natural to use, since related concepts tend to cluster together in the network. Similarly, an incompletely or inconsistently defined concept is easy to spot since a meaningful context is provided by those neighboring concepts to which it is already linked.

The analogy to hypertext is straightforward: Hypertext nodes can be thought of as representing single concepts or ideas, internode links as representing the semantic interdependencies among these ideas, and the process of building a hypertext network as a kind of informal knowledge engineering. The difference is that AI knowledge engineers are usually striving to build representations which can be mechanically interpreted, whereas the goal of the hypertext writer is often to capture an interwoven collection of ideas without regard to their machine interpretability. The work on semantic networks also suggests some natural extensions to hypertext, such as typed nodes, semistructured nodes (frames), and inheritance hierarchies of node and link types.

The advantages and uses of hypertext

Intertextual references are not new. The importance of hypertext is simply that references are machine-supported. Like hypertext, traditional literature is richly interlinked and is hierarchically organized. In traditional literature, the medium of print for the most part restricts the flow of reading to follow the flow of linearly arranged passages. However, the process of following side links is fundamental even in the medium of print. In fact, library and information science consist principally of the investigation of side links. Anyone who has done research knows that a considerable portion of that effort lies in obtaining referenced works, looking up cross-references, looking up terms in a dictionary or glossary, checking tables and figures, and making notes on notecards. Even in simple reading one is constantly negotiating references to other chapters or sections (via the table of contents or references embedded in the text), index entries, footnotes, bibliographic references, sidebars, figures, and tables. Often a text invites the reader to skip a section if he is not interested in greater technical detail.

But there are problems with the traditional methods.

- Most references can't be traced backwards: A reader can not easily find where a specific book or article is referenced in a document, nor can the author of a paper find out who has referenced the paper.
- As the reader winds his way down various reference trails, he must keep track of which documents he has visited and which he is done with.
- The reader must squeeze annotations into the margins or place them in a separate document.
- Finally, following a referential trail among paper documents requires substantial physical effort and delays, even if the reader is working at a well-stocked library. If the documents are on line, the job is easier and faster, but no less tedious.

New possibilities for authoring and design. Hypertext may offer new ways for authors and designers to work. Authoring is usually viewed as a word- and sentence-level activity. Clearly the word processor*

*Actually, the term "word processor" is quite misleading. Most such tools accept input only at the character level, and manipulate characters, words, sentences, and paragraphs with equal facility. So these tools manipulate units of text, not words. But do they "process" these units? "Processing" implies that the computer performs some additional work, such as changing the verb form if the subject was changed from singular to plural, or performing real-time spelling and grammar correction. Since this is not the case, we really should return to the original term for these tools: "text editors".

is a good tool for authoring at this level. However, authoring obviously has much to do with structuring of ideas, order of presentation, and conceptual exploration. Few authors simply sit down and pour out a finished text, and not all editing is just "wordsmithing" and polishing. In a broad sense, authoring is the *design of a document*. The unit of this level of authoring is the idea or concept, and this level of work can be effectively supported by hypertext, since the idea can be expressed in a node. As the writer thinks of new ideas, he can develop them in their own nodes, and then link them to existing ideas, or leave them isolated if it is too early to make such associations. The specialized refinements of a hypertext environment assist the movement from an unstructured network to the final polished document.

New possibilities for reading and retrieval. Hypertext may also offer new possibilities for accessing large or complex information sources. A linear (nonhypertext) document can only be easily read in the order in which the text flows in the book. The essential advantage of nonlinear text is the ability to organize text in different ways depending on differing viewpoints. Shasha provides the following description of this advantage:

> Suppose you are a tourist interested in visiting museums in a foreign city. You may be interested in visual arts. You may want to see museums in your local area. You may only be interested in inexpensive museums. You certainly want to make sure the museums you consider are open when you want to visit them. Now your guidebook may be arranged by subject, by name of museum, by location, and so on. The trouble is: if you are interested in any arrangement other than the one it uses, you may have to do a lot of searching. You are not likely to find all the visual arts museums in one section of a guidebook that has been organized by district. You may carry several guidebooks, each organized by a criterion you may be interested in. The number of such guidebooks is a measure of the need for a nonlinear text system.[21]

Another advantage is that it is quite natural in a hypertext environment to suspend reading temporarily along one line of investigation while one looks into some detail, example, or related topic. Bush described an appealing scenario in his 1945 article:

> The owner of the memex, let us say, is interested in the origin and properties of the bow and arrow. Specifically he is studying why the short Turkish bow was apparently superior to the English long bow in the skirmishes of the Crusades. He has dozens of possibly pertinent books and articles in his memex. First he runs through an encyclopedia, finds an interesting but sketchy article, leaves it projected. Next, in a history, he finds another pertinent item, and ties the two together. Thus he goes, building a trail of many items. Occasionally he inserts a comment of his own, either linking it into the main trail or joining it by a side trail to a particular item. When it becomes evident that the elastic properties of available materials had a great deal to do with the bow, he branches off on a side trail which takes him through textbooks on elasticity and tables of physical constants. He inserts a page of longhand analysis of his own. Thus he builds a [permanent] trail of his interest through the maze of materials available to him.[2]

As we have seen, Bush's notion of the "trail" was a feature of Trigg's Textnet,[6] allowing the hypertext author to establish a mostly linear path through the document(s). The main (default) trail is well marked, and the casual reader can read the text in that order without troubling with the side trails.

Summary. We can summarize the operational advantages of hypertext as:

- *ease of tracing references*: machine support for link tracing means that all references are equally easy to follow forward to their referent, or backward to their reference;
- *ease of creating new references*: users can grow their own networks, or simply annotate someone else's document with a comment (without changing the referenced document);
- *information structuring*: both hierarchical and nonhierarchical organizations can be imposed on unstructured information; even multiple hierarchies can organize the same material;
- *global views*: browsers provide table-of-contents style views, supporting easier restructuring of large or complex documents; global and local (node or page) views can be mixed effectively;
- *customized documents*: text segments can be threaded together in many ways, allowing the same document to serve multiple functions;
- *modularity of information*: since the same text segment can be referenced from several places, ideas can be expressed with less overlap and duplication;
- *consistency of information*: references are embedded in their text, and if the text is moved, even to another document, the link information still provides direct access to the reference;
- *task stacking*: the user is supported in having several paths of inquiry active and displayed on the screen at the same time, such that any given path can be unwound to the original task;
- *collaboration*: several authors can collaborate, with the document and comments about the document being tightly interwoven (the exploration of this feature has just begun).

The disadvantages of hypertext

There are two classes of problems with hypertext: problems with the current implementations and problems that seem to be endemic to hypertext. The problems in the first class include delays in the display of referenced material, restrictions on names and other properties of links, lack of browsers or deficiencies in browsers, etc. The following section outlines two problems that are more challenging than these implementation shortcomings, and that may in fact ultimately limit the usefulness of hypertext: *disorientation* and *cognitive overhead*.

Getting "lost in space." Along with the power to organize information much more complexly comes the problem of having to know (1) where you are in the network and (2) how to get to some other place that you know (or think) exists in the network. I call this the "disorientation problem." Of course, one also has a disorientation problem in traditional linear text documents, but in a linear text, the reader has only two options: He can search for the desired text earlier in the text or later in the text. Hypertext offers more degrees of freedom, more dimensions in which one can move, and hence a greater potential for the user to become lost or disoriented. In a network of 1000 nodes, information can easily become hard to find or even forgotten altogether. (See Figure 11.)

There are two major technological solutions for coping with disorientation—graphical browsers and query/search mechanisms. Browsers rely on the extremely highly developed visuospatial processing of the human visual system. By placing nodes and links in a two- or three-dimensional space, providing them with

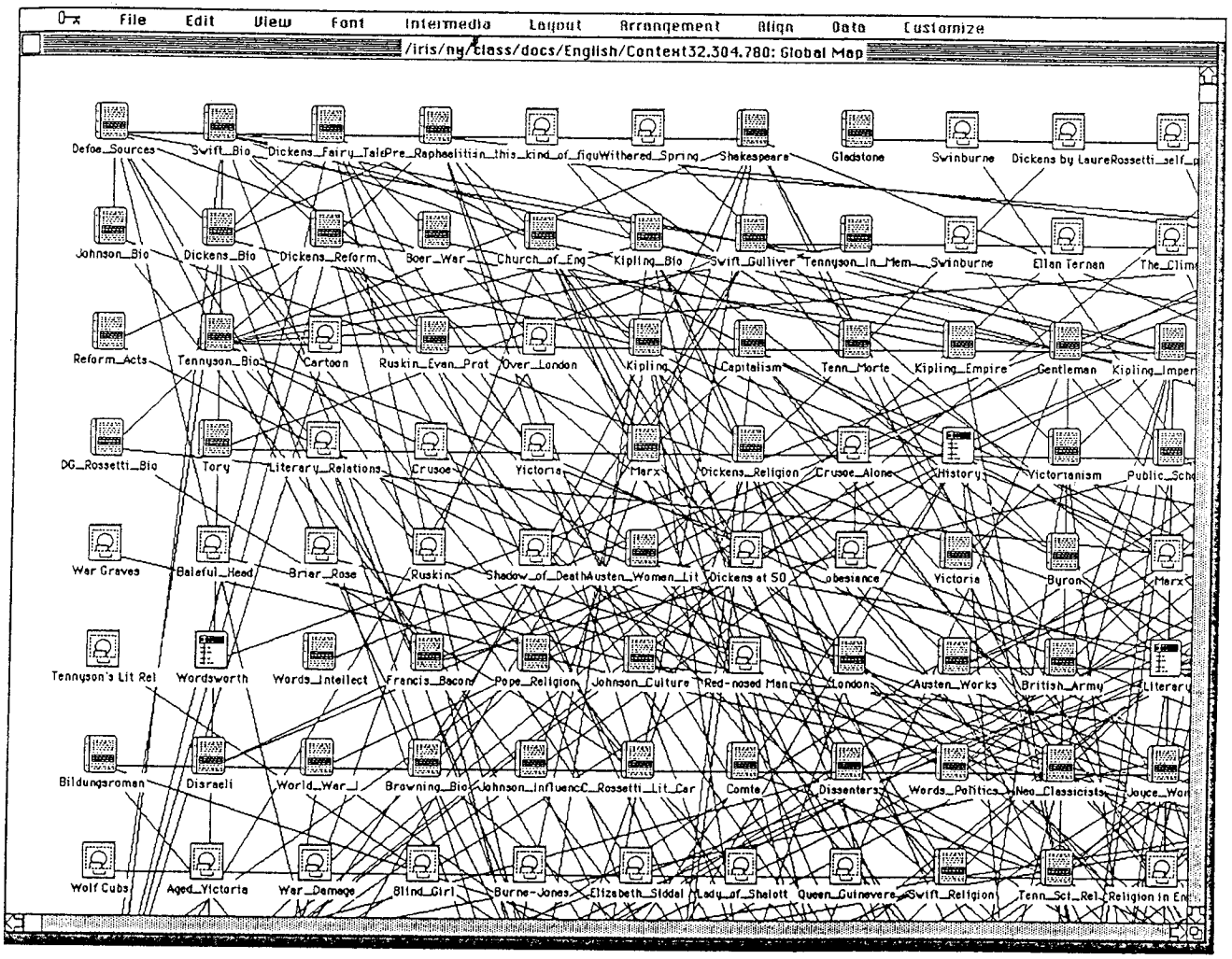


Figure 11. Tangled web of links. This experimental implementation of a global map in the Intermedia system shows the difficulty of providing users with spatial cues once a linked corpus contains more than a few dozen documents. This global map only represents about one tenth of the documents in a corpus designed for a survey of English literature course.

properties useful in visual differentiation (color, size, shape, texture), and maintaining certain similarities to our physical environment (for example, no two objects occupy the same space, things only move if moved, etc.), browser designers are able to create quite viable virtual spatial environments. Users orient themselves by visual cues, just as when they are walking or driving through a familiar city. However, there is no natural topology for an information space, except perhaps that higher level concepts go at the top or on the left side, so until one is familiar with a given large hyperdocument, one is by definition disoriented. In addition, an adequate virtuality is very difficult to maintain for a large or complex hypertext network. Such parameters as (1) large numbers of nodes, (2) large numbers of links, (3) frequent changes in the network, (4) slow or awkward response to user control inputs, (5) insufficient visual differentiation among nodes and/or links, and (6) nonvisually oriented users combine to make it practically impossible to abolish the disorientation problem with a browser alone.

One solution to this dilemma is to apply standard database search and query techniques to locating the node or nodes which the user is seeking. This is usually done by using boolean operations to apply some combination of keyword search, full string search, and logical predicates on other attributes (such as author, time of creation, type, etc.) of nodes or links. Similarly, one can filter (or *ellide*) information so that the user is presented with a manageable level of complexity and detail, and can shift the view or the detail suppression while navigating through the network. However, much research remains to be

done on effective and standardized methods for ellision.

The cognitive task scheduling problem. The other fundamental problem with using hypertext is that it is difficult to become accustomed to the additional mental overhead required to create, name, and keep track of links. I call this "cognitive overhead." Suppose you are writing about *X*, and a related thought about *Y* comes to mind and seems important enough to capture. Ideally, hypertext allows you to simply "press a button" (using some mouse or keyboard action) and a new, empty hypertext window pops onto the screen. You record *Y* in this new window, then you press another button, the *Y* window disappears, and you are in the *X* window right where you were when *Y* occurred to you.

Unfortunately, the situation is a bit more complex than this scenario implies. If *Y* has just occurred to you, it may still be hazy and tentative; the smallest interruption could cause you to lose it. Coming up with a good word or short phrase to summarize *Y* may not be easy. You have to consider not just what is descriptive but also what will be suggestive for the reader when he encounters the link to *Y* within *X*. In addition, you must determine whether you should name the link to *Y* to suggest the contents of *Y* or to show *Y*'s relationship to *X*. Some systems (for example, NoteCards) provide that links can have both a *type* (such as "idea") and a *label* (such as "subsume *A* in *B*"). Coming up with good names for both can impose even more load on an author struggling with an uncertain point. (One way to reduce this problem is for the authoring system to support immediate recording of the substance of the idea, deferring the creation and labeling of the link and/or the node until after the thought has been captured.)

Beyond that, you must also consider if you have provided sufficient links to *Y* before returning to work on *X*. Perhaps there are better ways to link *Y* to the network of thoughts than at the point in *X* where *Y* came to mind.

The problem of cognitive overhead also occurs in the process of reading hypertext, which tends 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 metalevel decision making, an overhead that is absent when the author has already made many of these choices for you. At the moment that you encounter a link, how do you decide if following the side path is worth the distraction? Does the label appearing in the link tell you enough to decide? This dilemma could be called "informational myopia." The problem is that, even if the system response time is instantaneous (which it rarely is), you experience a definite distraction, a "cognitive loading," when you pause to consider whether to pursue the side path. This problem can be eased by (1) having the cross-referenced node appear very rapidly (which is the approach of KMS), (2) providing an instantaneous one- to three-line explanation of the side reference in a pop-up window (which is the approach of Intermedia), and (3) having a graphical browser which shows the local subnetwork into which the link leads.

These problems are not new with hypertext, nor are they mere byproducts of computer-supported work. People who think for a living—writers, scientists, artists, designers, etc.—must contend with the fact that the brain can create ideas faster than the hand can write them or the mouth can speak them. There is always a balance between refining the current idea, returning to a previous idea to refine it, and attending to any of the vague "proto-ideas" which are hovering at the edge of consciousness. Hypertext simply offers a sufficiently sophisticated "pencil" to begin to engage the richness, variety, and interrelatedness of creative thought. This aspect of hypertext has advantages when this richness is needed and drawbacks when it is not.

To summarize, then, the problems with hypertext are

- *disorientation*: the tendency to lose one's sense of location and direction in a nonlinear document; and
- *cognitive overhead:* the additional effort and concentration necessary to maintain several tasks or trails at one time.

These problems may be at least partially resolvable through improvements in performance and interface design of hypertext systems, and through research on information filtering techniques.

In this article, I have reviewed existing hypertext systems, the opportunities and problems of hypertext, and some of the top-level design issues of building hypertext systems. It has been my intention to give the reader a clear sense of what hypertext is, what its strengths and weaknesses are, and what it can be used for. But I also intended something more: that the reader come away from this article excited, eager to try using hypertext for himself, and aware that he is at the beginning of something big, something like the invention of the wheel, but something that still has enough rough edges that no one is really sure that it will fulfill its promise.

To that end, I mention one more book that might be considered to belong to the literature on hypertext. *Neuromancer*[24] is a novel about a time in the distant future when the ultimate computer interface has been perfected: One simply plugs one's brain into the machine and experiences the computer data directly as perceptual entities. Other computers look like boxes floating in three-dimensional space, and passwords appear as various kinds of doors and locks. The user is completely immersed in a virtual world, the "operating system," and can move around and take different forms simply by willing it.

This is the ultimate hypertext system. The basic idea of hypertext, after all, is that ideas correspond to perceptual objects, and one manipulates ideas and their relationships by directly manipulating windows and icons. Current technology limits the representation of these objects to static boxes on a CRT screen, but one can easily predict that advances in animation, color, 3D displays, sound, etc.—in short, Nelson's *hypermedia*—will keep making the display more active and realistic, the data represented richer and more detailed, and the input more natural and direct. Thus, hypertext, far from being an end in itself, is just a crude first step toward the time when the computer is a direct and powerful extension of the human mind, just as Vannevar Bush envisioned when he introduced his Memex four decades ago.□

Acknowledgements

I wish to thank Les Belady, Bill Curtis, Susan Gerhart, Raymonde Guindon, Eric Gullichsen, Frank Halasz, Peter Marks, and Andy van Dam for their thoughtful reading of previous drafts.

References

1. T.H. Nelson, "Getting It Out of Our System," *Information Retrieval: A Critical Review*, G. Schechter, ed., Thompson Books, Wash., D.C., 1967.
2. V. Bush, "As We May Think," *Atlantic Monthly*, July 1945, pp.101-108.
3. D.C. Engelbart, "A Conceptual Framework for the Augmentation of Man's

Intellect," in *Vistas in Information Handling*, Vol. 1, Spartan Books, London, 1963.

4. D.C. Engelbart and W.K. English, "A Research Center for Augmenting Human Intellect," *AFIPS Conf. Proc.*, Vol. 33, Part 1, The Thompson Book Company, Washington, D.C., 1968.
5. T.H. Nelson, "Replacing the Printed Word: A Complete Literary System," *IFIP Proc.*, October 1980, pp. 1013-1023.
6. R.H. Trigg, *A Network-based Approach to Text Handling for the Online Scientific Community*, PhD. Thesis, University of Maryland, 1983.
7. H. Rittel and M. Webber, "Dilemmas in a General Theory of Planning," *Policy Sciences*, Vol. 4, 1973.
8. D.G. Lowe, "Cooperative Structuring of Information: The Representation of Reasoning and Debate," in *Int'l. J. of Man-Machine Studies*," Vol. 23, 1985, pp. 97-111.
9. J.B Smith et al, "WE: A Writing Environment for Professionals," Technical Report 86-025, Department of Computer Science, University of North Carolina at Chapel Hill, August 1986.
10. W. Hershey, "Idea Processors," *BYTE*, June 1985, p. 337.
11. D. McCracken and R.M. Akscyn, "Experience with the ZOG Human-computer Interface System," *Int'l J. of Man-Machine Studies*, Vol. 21, 1984, pp. 293-310.
12. B. Shneiderman and J. Morariu, "The Interactive Encyclopedia System (TIES)," Department of Computer Science, University of Maryland, College Park, MD 20742, June 1986.
13. J.H. Walker, "The Document Examiner," *SIGGRAPH Video Review*, Edited Compilation from *CHI'85: Human Factors in Computing System*, 1985.
14. F.G. Halasz, T.P. Moran, and T.H. Trigg, "NoteCards in a Nutshell," *Proc. of the ACM Conf. on Human Factors in Computing Systems*, Toronto, Canada, April 1987.
15. N.L. Garrett, K.E. Smith, and N. Meyrowitz, "Intermedia: Issues, Strategies, and Tactics in the Design of a Hypermedia Document System," in *Proc. Conf. on Computer-Supported Cooperative Work*, MCC Software Technology Program, Austin, Texas, 1986.
16. N. Yankelovich, N. Meyrowitz, and A. van Dam, "Reading and Writing the Electronic Book," *Computer*, October 1985.
17. N. Delisle and M. Schwartz, "Neptune: A Hypertext System for CAD Applications," *Proc. of ACM SIGMOD Int'l Conf. on Management of Data*, Washington, D.C., May 28-30, 1986, pp. 132-143. (Also available as SIGMOD Record Vol. 15, No. 2, June 1986).
18. A. diSessa, "A Principled Design for an Integrated Computational Environment," *Human-Computer Interaction*, Vol. 1, Lawrence Erlbaum, 1985, pp. 1-47.
19. K.M.Pitman, "CREF: An Editing Facility for Managing Structured Text," A.I. Memo No. 829, M.I.T. A.I. Laboratory, Cambridge, Mass., February 1985.
20. P.J. Brown, "Interactive Documentation," in *Software: Practice and Experience*, March 1986, pp. 291-299.
21. D. Shasha, "When Does Non-Linear Text Help? *Expert Database Systems, Proc. of the First Int'l Conf.*, April 1986, pp. 109-121.
22. A. Kay and A. Goldberg, "Personal Dynamic Media," *Computer*, March 1977, pp. 31-41.
23. T.W. Malone et al, "Intelligent Information-Sharing Systems," *Communications of the ACM*, May 1987, pp. 390-402.
24. W.Gibson, *Neuromancer*, Ace Science Fiction, 1984.

A more detailed version of this article, including an extended bibliography, is available from the author. To obtain a copy, circle number 181 on the Reader Service Card at the back of the magazine.

E. Jeffrey Conklin is a member of the research staff and GE's liaison to the Software Technology Program in the Microelectronics and Computer Technology Corporation (MCC). His research centers on constructing information systems for the capture and use of design rationale.

Conklin Received his BA from Antioch College and his MS and PhD from the University of Massachusetts at Amherst.

Readers may write to Conklin at MCC Software Technology Program, P.O. Box 200195, Austin, TX 78720; (512) 343-0978.