

A Comparison
of
Hypertext Systems

John L. Schnase
John Leggett
Charles J. Kacmar
Craig Boyle

Hypermedia Research Lab

SEPTEMBER 1988
TAMU-HRL 88-017

Table of Contents

Introduction	1
Hypertext System Architecture	2
Definition of Terms	4
Definition of Hypertext System Functions	8
Minimum Hypertext System Functionality	12
Characteristics of Selected Hypertext Systems	14
KMS	15
NoteCards	19
Intermedia	22
Augment	25
Neptune / HAM	27
HyperTIES	30
Guide	33
HyperCard	36
Concordia / Document Examiner	38
VisiDoc	42
Tabular Comparisons of Selected Hypertext Systems	44
Literature Cited	50

A Comparison of Hypertext Systems

John L. Schnase

John Leggett

Charles J. Kacmar

Craig Boyle

Introduction

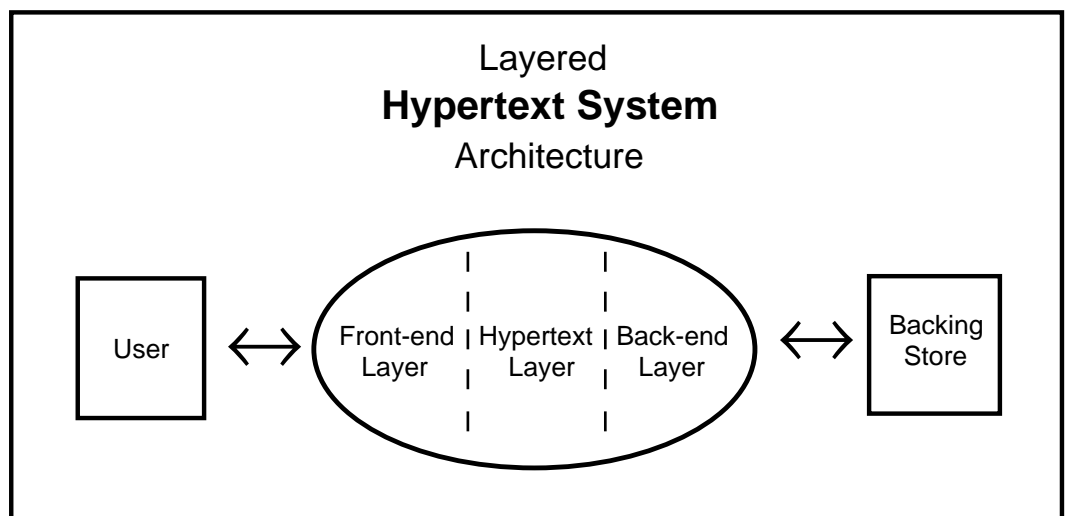
This paper presents the functional characteristics of eight hypertext systems. While many claim hypertext functionality, the systems chosen for this analysis are significant in the extent to which their design has shaped our understanding and expectations about hypertext systems. They are also, for the most part, among the more accessible, complete, and well documented of the systems presently available.

We will describe KMS, NoteCards, Intermedia, Augment, Neptune/HAM, HyperTIES, Guide, and HyperCard. In addition, we will look at two documentation systems that have a significant amount of hypertext functionality, Visidoc and Concordia/Document Examiner. To clarify our presentation, we first introduce a framework for discussing hypertext system architecture, define terminology, and identify functions commonly provided by these systems, including a minimum set of functions required of all hypertext systems. Important characteristics of each system are then presented, along with comments on usability and performance. A tabular comparison of system functionality is also provided.

There is growing recognition among architects and implementors of hypertext systems of the need for a uniform terminology to describe major concepts in this area. In addition to providing a functional overview, it is hoped that this paper contributes to a clearer understanding of the design rationale behind major, existing hypertext systems and encourages greater care in developing a vocabulary to describe hypertext.

Hypertext System Architecture

We propose the layered hypertext system architecture as a conceptual framework for organizing discussions of hypertext system functionality and design. As shown below, a hypertext system is assumed to be composed of three layers of functionality: front-end, hypertext, and back-end layers. Each function present in a system (See "Definition of Hypertext System Functionality") may be categorized as being a front-end, hypertext, or back-end function. Typical front-end layer functions include display operations and capturing user input; some back-end functions are retrieving, storing, and caching. The hypertext layer implements the functionality and data abstractions required to translate user requests at the front-end into actions in the back-end.



In most cases, one would expect functions in the various categories to be implemented in their respective layers. However, boundaries between the three layers are not fixed, and in the design of a hypertext system, there may be an option of implementing a function in either of two adjoining layers. For example, caching is a back-end function that could be implemented in the back-end layer of one system without using data structures or operations present in the hypertext layer. In another system, caching might rely heavily on functionality present in the hypertext layer, in which case this back-end function would be viewed as implemented in the hypertext layer. It is also

important to note the extent to which systems implement functionality on their own, or draw upon that which is already present in an operating, windowing, or database system.

These distinctions may, at first, appear to be of little value, and the designers of existing hypertext systems, for the most part, have not explicitly implemented a layered model. Two notable exceptions are Xanadu [Gregory 1983] and HAM [Campbell and Goodman 1988] which are essentially hypertext system back-ends. For a more complete description of these systems see Kacmar *et. al.* 1988. Presumably, any hypertext layer placed on top of Xanadu or HAM could invoke back-end functionality directly from these modules. The greatest utility of the proposed architecture is the structure it will bring to future hypertext system design.

Definition of Terms

A new vocabulary is emerging along with the increased interest in hypertext. As is often the case, there is presently no consensus on the precise meaning of several words used to describe hypertext and hypertext system functions. This section provides definitions for terms that are particularly relevant to a comparison of hypertext system characteristics. In developing these definitions, emphasis is placed on identifying, whenever possible, the creator's intent when the term was originally applied, capturing its most general meaning, and, when appropriate, commenting on confusions that have arisen by uses of the term. In addition, we provide new definitions or introduce our own terminology when doing so clarifies a concept. The present list is by no means complete, and it is assumed that increasing knowledge about hypertext will lead to greater clarity and rigor in its terminology.

Hyper-. "... as used here connotes extension and generality. The criterion for this prefix is the inability of ... objects to be comprised sensibly into linear media" [Nelson 1965].

Hypertext. "... a body of written or pictorial material interconnected in such a complex way that it could not conveniently be presented or represented on paper. [Hypertext] is useful where relationships are unclear; where contingencies and tasks are undefined and unpredictable; where the structures or final outcome it must represent are not yet fully known; ... or where things are in permanent and unpredictable flux; [... where] as in biology, the only ultimate structure is change itself." [Nelson 1965].

Hypermedia. "Films, sound recordings, and video recordings are also linear strings. But these too can ... be arranged as non-linear systems ... for editing purposes, or for display with different emphasis. The hyperfilm [... for example] is only one of the possible hypermedia that require our attention." [Nelson 1965]. "Presumably hypertexts may contain graphics of various kinds" [Nelson 1967b]. The term hypermedia, as it is commonly used, implies that information may be present in systems in forms other than text. It appears that Nelson intended "hypertext" to be sufficient to describe the range of media these systems would eventually manage.

Hypertext system. A functionally related set of computer hardware and software components that provide support for authoring and browsing hypertext. (Also see the section entitled "Minimum Hypertext System Functionality.")

Author. The component of a hypertext system that creates hypertext.

Browser. The component of a hypertext system that allows information contained in hypertext to be accessed.

Node. The container for information in a hypertext. The concept of a node being a chunk of text in hypertext was introduced by Nelson [1967a] in an analogy he drew between connected information in hypertext and the use of arcs and nodes to represent associations in graph theory. Engelbart *et. al.* [1973] adopted Nelson's use of the term and it has subsequently made its way into the hypertext vocabulary. Nodes may contain text, graphics, sound, etc.

It is difficult to develop a precise definition for the term node and important misunderstandings relate to its varied interpretation. It is clear that Bush [1945], Nelson [1967a], and Engelbart *et. al.* [1973] intend nodes to represent associated information. However, most designers of hypertext systems implement a concept of node as a unit of storage or as a unit for display without distinguishing between information contained in a node and the node as a container for information. A distinguishing feature of hypertext is interconnected information -- a web of ideas that is in some sense seamless. Implementation concerns force hypertext systems to be aware of presentational and storage breaks in this information web. In order to compare hypertext system functionality, it appears beneficial to define node with respect to each system's implementation, or model, and remember to draw a careful conceptual distinction between node and the information it contains.

Node size. Node size is generally expressed as a unit for storage or as a unit for display. If the maximum size of a node is based on a unit for storage that exceeds the presentational capacity of the physical display, then facilities for paging or scrolling are implied.

Typed node. Nodes may be typed in order to define a set of data structures and valid operations on nodes of a given type. A typed node is a member of a class of nodes.

Named node. A character string may be associated with nodes in order to provide organizational assistance to the user, facilitate searching, or facilitate filtering.

Composite node. A node containing one or more "include" or "automatic include" links. Composition allows nodes to be created that contain elements contained in other nodes.

Anchor. The source or destination of a link. Presumably, an anchor is tied to information that is associated by links to other information in the hypertext. Anchors are commonly characters, words, pixels, graphics, or nodes.

Link. A connector between two anchors. Existing hypertext systems typically provide support for unidirectional links. The following list describes the commonly implemented forms of links:

- **"to" link.** A connector whose source anchor is an element contained in a node and whose destination anchor is a node. A link from content to structure. Following a "to" link causes what is presently displayed to be replaced by the contents of the destination node.

- **"into" link.** A connector whose source and destination anchors are elements contained in a node. A link from content to content. Following an "into" link causes what is presently displayed to be replaced by elements constituting the destination anchor.

- **"back" link.** The ability to follow any link that points to or into a node as if it were bi-directional.

- **"include" link.** A "to" or "into" link which when followed causes a portion of the current display to be modified rather than replaced by elements from the destination node. Implies node composition.

- **"automatic include" link.** An "include" link that the system follows automatically when the source anchor is brought into the display. Implies node composition.

- **"stretch" link.** A form of "include" link that is used to connect stretchtext.

Typed link. Links may be typed in order to define a set of data structures and valid operations on links of a given type. A typed link is a member of a class of links.

Named link. A character string may be associated with links in order to provide organizational assistance to the user, facilitate searching, or facilitate filtering.

Extensible. The functionality of a hypertext system may be extended by the user through use of a programming language that is integrated with the system.

Collateral display. The ability to display the contents of nodes side-by-side in tiled windows. This is especially useful for comparing versions.

View. The elements of a hypertext that are revealed to a user. Specifically, the nodes and contents of nodes that may be displayed, and the anchors and links that may be selected by a user. Some systems allow differing views of the same hypertext.

Filter. The process of generating a view.

Stretchtext. A form of hypertext consisting of "ordinary continuous text that can be 'stretched', or made ... more detailed" [Nelson 1967c]. It is implemented using "stretch" links where the information brought into the display does not replace or overlay what is presently displayed, but separates it and becomes inserted. In addition, the information obtained when a "stretch" link anchor is selected is composed so that a continuous narrative results. Selected stretchtext may be "unstretched," returning the user to a previous, less detailed context. "Thus it is unlike conventional hypertexts having discrete chunks and breaks ... " [Nelson 1967c] .

Definition of Hypertext System Functions

The functionality present in a hypertext system and the manner it is provided to the user varies considerably from system to system. However, it is possible to identify several essential features common to all complete hypertext systems. This section provides a basic vocabulary for discussing system functionality. Here again, a major motivation is to create definitions that specifically facilitate a comparison of the functionality provided by existing systems. In developing these definitions, an effort has been made to categorize each function as being primarily author or browser functions. Also, whenever appropriate, we have tried to place the functionality within the conceptual framework of a layered hypertext system by identifying where it reasonably should reside among front-end, hypertext, and back-end layers.

Create/delete. The process of creating and deleting anchors, links, and nodes involves functionality present in front-end, hypertext, and back-end layers of a hypertext system. The hypertext layer communicates with the front-end to capture the user's intent and record their input as they create or modify the contents of a node. The hypertext layer then passes information to the back-end layer which invokes functions, such as create, delete, lock, and store, required to incorporate the input into hypertext. Creating and deleting are author functions.

- **create/delete anchor.** Specify that a group of characters, words, pixels, or a graphical image, etc., is to become a source or destination anchor for a link. The ability to direct removal of an existing anchor.

- **create/delete link.** Select source and destination anchors and direct that a link be forged between them. Remove existing links. The functions of creating anchors and creating the associated links is provided as a single operation in many systems. Others provide separate steps for creating anchors and links, thus allowing decisions about how best to link candidate anchors to be delayed. Some of the current hypertext systems allow more than one link to originate from a single source anchor.

- **create/delete node.** Specify that the input gathered by the front-end and hypertext layer become the contents of a new node in a hypertext. Delete an existing node.

Display. Displaying nodes, links, and anchors are front-end hypertext system functions common to both author and browser. The hypertext layer cooperates with the front-end by managing the internal data structures required for display.

- **display node.** Present the contents of a selected node on a physical display.

- **display anchor.** Distinguish anchors in the displayed node contents. This is often done by highlighting, underscoring, coloring, boxing, or marking an anchor with a special symbol.

- **display link.** In systems that allow multiple links to originate from the same anchor, the link options that are available at a selected anchor must be displayed.

Select. Select operations are browser functions that draw on functionality present in all three layers of a hypertext system. Capturing the user's intent requires cooperation between the front-end and hypertext layer. Actually retrieving the destination anchor is a back-end function invoked by the hypertext layer. Finally, the destination anchor is displayed by the front-end as mediated by the hypertext layer.

- **select anchor.** Identify an anchor and direct the system to follow its associated link to display the destination anchor.

- **select link.** In systems that allow multiple links to originate from one anchor, selecting an anchor becomes a two-step operation. First the anchor is selected and the link options are displayed, then a link must be selected which the system follows to retrieve the destination anchor.

Search. Searching is primarily a back-end function associated with the browser component of a hypertext system.

- **content search.** The content of nodes in the hypertext are searched for a match to a given query, for example, "all nodes containing 'INTERTWINGLED'"

- **structure search.** Examines subnetworks of nodes within a hypertext searching for a given pattern, for example, "all nodes of Type X that are connected to nodes of Type Y"

- **spatial search.** Searches nodes in the hypertext for a match on a spatial query, for example, "two columns of text with a picture in the upper right-hand corner"

Copy. Copying operations are primarily back-end functions associated with the author.

- **copy a node.** Copy the contents of a node and all associated anchor and link information.

- **copy node contents.** Copy only the contents of a node without any of the associated anchor and link information.

Import. Allow the contents of a file to be brought into a node and incorporated into a hypertext. An author function that resides in the hypertext and back-end layers.

Export. Allow the contents of a node to be extracted from a hypertext and stored in a conventional file. This is a hypertext and back-end layer function, but it is difficult to categorize export as being an author or browser function. Export is probably not an operation that should be performed often on information stored in a hypertext.

Version. Maintain a history of changes to the contents of a node or set of nodes. A hypertext and back-end layer function associated with the author component of a hypertext system.

Print. Format and print the contents of a node. A hypertext layer function that cannot easily be assigned to either author or browser. Printing is probably not an operation that should be performed often on information stored in a hypertext.

History. A record of the destination anchors accessed by a user while browsing hypertext. A browser function implemented in the hypertext layer.

Overview. A graphical representation of nodes in a hypertext. A (graphical) browser function implemented in the front-end and hypertext layer.

Retrieve. Retrieve a node from backing store, including its contents, anchor, and link information. A back-end function associated with author and browser.

Store. Store a node, including its contents, anchor, and link information, on backing store. A back-end function associated with the author.

Index. Maintain information about the hypertext. For example, an index of all nodes that contain graphical objects. A back-end function associated with author and browser.

Cache. Move nodes from backing store into main memory in order to reduce access time. Does not necessarily imply that cache memory is being used. A browser function implemented in hypertext or back-end layer.

Lock. Enforce singular write access to nodes, links, and anchors in a shared environment. A back-end function important to authoring.

Protection. Setting access rights on nodes or performing encryption and decryption. A back-end function that affects author and browser.

Backup and recovery. Protect information by saving and restoring copies of the hypertext. A back-end function important to author and browser.

Shared access. Allowing more than one user to access hypertext or the components of hypertext, such as anchors, links and nodes, at different times.

Concurrent access. Allowing more than one user to access hypertext or the components of hypertext at the same time.

Distributed access. Allowing access to hypertext or the components of hypertext residing on more than one machine.

Minimum Hypertext System Functionality

In this section we identify the minimum functionality required for a system to qualify as a hypertext system. In doing so we apply a more rigid standard than some system designers would like. We feel that it is useful to view the development of hypertext systems as an evolutionary process still in its earliest stages.

Bush [1945], Engelbart [1962], and Nelson [1965] were among the first to propose systems that augment the human intellect by managing information stored in what we now call hypertext. It is clear from the literature that these early proponents had a strong sense that the connections in hypertext were linked ideas, that associations could be easily and arbitrarily forged, and that the information could be personalized, freely annotated, freely viewed and readily accessed. They proposed systems that offer a direct approach to data management and rely heavily on the user's increased use of visual cues, spatial reasoning, and associative thought. The revolutionary content of their ideas was, and continues to be, the extent to which these systems engage the user as an active participant in interactions with information.

Many information management systems are presently available that purport to be hypertext systems. Clearly, most still fall short of the pioneers' vision. This is not surprising since much of the technology required for the present systems only recently became available. In order to understand the functionality provided by current hypertext systems and draw comparisons among them, it is useful to define a minimum set of functions required of a hypertext system. Hypertext systems may then be viewed as existing on a continuum that ranges from minimal to increasingly sophisticated.

The minimum functionality required in order for a system to qualify as a hypertext system includes the ability to create, delete, store, and retrieve nodes, links, and anchors; display nodes and anchors; select anchors; and, in a shared environment, lock nodes, links, and anchors. In addition to these functional criteria, an important distinguishing feature of a hypertext system -- a quality that can at times be difficult to assess -- is the degree to which the system accommodates "[complex interconnections] that could not conveniently be presented or represented on paper" [Nelson 1965].

It is often convenient to group these criteria into author and browser functions. An author must be able to create, delete, store, retrieve, lock and display; a browser must be able to retrieve, display,

and select. A complete hypertext system must, therefore, contain both author and browser. Neither Visidoc nor Document Examiner are hypertext systems because they lack an authoring component. It would be more accurate to describe them as hypertext browsers. Likewise, Concordia alone is an authoring system with hypertext functionality, not a hypertext system. It may also be helpful, at times, to categorize systems with respect to the layered hypertext system architecture. For example, Xanadu and HAM are essentially hypertext system back-ends [Kacmar *et. al.* 1988]. The table below presents the systems examined in this paper and their classification.

Finally, it is appropriate to comment on why we do not accept the combination of Concordia and Document Examiner as a hypertext system. It is clear from the descriptions that follow that these systems are strongly biased towards the production of linear, paper-based documentation. The designers of Concordia/Document Examiner included a significant amount of hypertext system functionality, but refrain from labeling their system "hypertext." This is probably because their design emphasis was not strongly oriented towards arbitrarily-linked, non-linear, electronic information -- information not easily "presented or represented" on paper. Even the combined system fails to meet this last criterion in the definition of a hypertext system.

As we see in the following analysis, the functionality provided by these systems and their suitability to specific applications varies considerably. It is unlikely that further classification would be helpful at the present time.

System	Classification
KMS	Hypertext System
NoteCards	Hypertext System
Intermedia	Hypertext System
Augment	Hypertext System
Neptune /	Hypertext System
HAM	Hypertext Back-end
HyperTIES	Hypertext System
Guide	Hypertext System
HyperCard	Hypertext System
Concordia /	Authoring System
Document Examiner	Hypertext Browser
Visidoc	Hypertext Browser

Characteristics of Selected Hypertext Systems

The following section provides a description of the most important characteristics of eight hypertext systems: KMS, NoteCards, Intermedia, Augment, Neptune/HAM, HyperTIES, Guide, and HyperCard. In addition, we provide information on two documentation systems, Concordia/Document Examiner and VisiDoc, that incorporate a significant amount of hypertext system functionality.

For each system we briefly remark on its development history, intended audience, historical use, architectural requirements, and future prospects. We then describe its data model and underlying metaphor. Nodes, links, authoring, and browsing are described next. Finally, we comment, whenever appropriate, on our impressions about usability and performance, and close by giving the most important references on the system.

It should be pointed out that it is often difficult to characterize the performance and usability of these systems. In many cases performance has less to do with the hypertext system itself than the platform on which it runs or specific network characteristics of its environment. Usability is often closely related to the types of problems being solved or the information being managed by a system. Assessments of usability and performance are likely to remain qualitative until more formal standards emerge.

In describing these systems, we attempt to provide sufficient detail to communicate their most important design features. What follows is not a collection of detailed user's manuals, but system summaries that are a reasonable starting point for further reading. For a more detailed discussion of the back-end functionality of each of these systems see *Data Management Facilities of Existing Hypertext Systems* by Kacmar *et. al.* [1988].

The Hypertext Research Lab at Texas A&M University has worked first-hand with all but three of the systems in this report. Intermedia, Augment and Neptune/HAM are not available in our lab; however, individuals on the research team have had limited exposure to these systems. The systems that are available have been used extensively over the past several months.

KMS

KMS (Knowledge Management System) is a distributed hypertext system for workstations. The system was developed by Robert Akscyn, Donald McCracken, and Elise Yoder at Knowledge Systems, Inc., and grew out of their previous experiences with the ZOG Project at Carnegie-Mellon University. KMS has been under development since 1981. Release 6A is scheduled to be available in August 1988. This analysis reports on KMS - Release 5B.

KMS supports organization-wide collaboration for a broad range of applications, including electronic publishing, on-line documentation, project management, software engineering, and computer-aided instruction. In addition, KMS has been used for financial modeling, accounting, and as a user interface to minidisk-based materials and expert systems. The designers are shaping KMS to exploit what they believe will be the dominant architecture for organizational computing environments of the 1990's: wide-area networks of large-screen, diskless workstations. KMS presently runs on Sun and Apollo workstations [Akscyn *et. al.* 1988].

Data model and underlying metaphor. A KMS database consists of a set of interlinked, screen-sized workspaces called frames. Groups of related frames may be organized into framesets.

Although KMS databases may have any structure their creators desire, most have a strong hierarchical orientation. There is usually a multilevel hierarchy that acts as a skeleton for the entire database, and top levels of the hierarchy serve primarily as indexes to the entire database.

The central KMS metaphor is a universe of connected spaces through which users rapidly travel, like pilots navigating spacecraft in the real universe. Users navigate from frame to frame by pointing the mouse cursor at an item linked to another frame and clicking a button. KMS accesses the linked frame and displays it within the same window.

Nodes. A node, or frame, in KMS is what may be displayed on a single screen. A frame may contain any number of objects, each of which may be positioned independently. Objects may be text, graphics, images (bit mapped and PostScript generated), or points and may be connected to form complex structures. A property list is associated with a frame and each object it contains. This list carries such information as link destination, font characteristics, and actions. Frames are named automatically by the system when they are created; however, frames are not typed.

Frames are stored as Unix files and are placed into framesets. A frameset is a Unix directory. The maximum number of frames in a KMS database is limited only by available disk space. KMS supports shared, concurrent, and distributed access to frames. Concurrent updates to shared frames are handled by alerting the user that their current frame's backing store version has been updated since its last retrieve, then saving the current frame's contents in a new frame. Protection may be specified on frames from within KMS, but KMS's internal protection does not, as yet, map to Unix file protection.

Links. Source anchors in KMS frames may be text or points. Points are usually associated with graphical objects. Destination anchors are frames. A single, untyped, unnamed "to" link may be associated with a source anchor along with an action to be invoked when the link is followed. All valid source anchors in a frame may be linked to other frames. Information about the links present in a frame is stored along with the frame contents.

Authoring. In KMS there is no mode boundary between authoring and browsing; a user can directly manipulate the contents of a frame at any time. A major design goal for this system has been improved performance through user interface simplicity. Most common operations may be invoked through a context-sensitive mouse cursor that provides a variety of commands depending on screen location. Other navigation functions and utilities are easily accessed from a command line at the bottom of the screen.

To build links, the source anchor is selected by the mouse cursor, then a link is established by either automatically creating a new frame or, if it is a link to an existing frame, updating the anchor's property list with the destination frame's name. Anchors are specified at the time links are created. Text and graphics may be imported and exported, versioning is supported (Release 6A), and an action language provides extensibility. KMS offers an array of editing features, fonts, and stylesheets useful for creating documents. A "linearize" program formats a tree of frames for printing on PostScript-compatible printers by traversing "to" links in a depth-first manner treating the links as "include" links. The tree may cross frameset boundaries.

The most important characteristic of a KMS frame is its spatial nature. Like space in the real world, space in a frame "exists" whether or not any objects occupy it. Space in a frame provides a

background on which objects may be repositioned and provides a natural command context -- when the cursor is in empty space, the user can create points, lines, text, etc. More importantly, empty space on a frame allows the user to annotate the contents of a frame.

Browsing. A user may display and print the entire contents of a single, full-screen frame. However, the workstation screen is normally split into two tiled windows, each of which shows the left half of a frame. This size is sufficient for one page worth of material, and KMS's "linearize" program only prepares this half for printing. The right half of a frame may be used for annotations. Frames displayed in parallel are in no way related, that is, each window in a split display provides a separate thread of navigation through the KMS database.

Source anchors are distinguished by cursor sensitivity and small, hollow bullets in front of linked items. Anchors are selected by moving the cursor to the anchor and clicking the left mouse button. The currently displayed frame is then replaced by the destination frame. KMS maintains back pointers on a stack, and the user may navigate backwards through previously selected frames by clicking on empty space. Alternatively, the user may go directly to any frame in any frameset. Histories are not displayed, and no graphical overview is provided.

An index is not directly provided; however, KMS's search facility will generate an index frame containing candidate frame titles that result from a content search of the frames in a frameset. This provides a limited view facility.

Usability and performance. The designers of KMS believe the ability to browse quickly in a hypertext system is critical to its usability, especially for large-scale databases. System response time for accessing and displaying frames on Sun 386i Model 250 Workstations ranges from 0.12 seconds for a small, cached frame to 1.46 seconds for a large frame accessed from remote disk. Average access time is approximately 0.5 seconds [Akscyn *et. al.* 1988]. In addition, the visual regularity of a standard frame layout, anchors that provide relatively large targets for selection, fast backtracking, and no scrolling all assist the user in maintaining orientation while navigating the hypertext. KMS is appropriate for a wide range of applications.

References.

Akscyn, R. M., D. L. McCracken, and E. A. Yoder. 1988. KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations. *Communications of the ACM*, Vol. 31, No. 7, (July), 820-835.

KMS Reference Manual. 1987. Knowledge Systems, Inc., Murrysville, PA.

NoteCards

NoteCards is a general hypertext system for workstations. The system was developed by Randall Trigg, Thomas Moran, and Frank Halasz at Xerox PARC as a research vehicle. NoteCards has been available since 1985, but there has never been an intense effort to develop it as a commercial product. It is still under development and a new release is expected soon that would, among other things, support a distributed environment. To date, NoteCards is available for a range of Xerox Lisp computers. This analysis reports on NoteCards - Release 1.2K running on a Xerox 1108.

NoteCards was designed to help people work with ideas. Its intended users are authors, researchers, designers, and other intellectual laborers engaged in analyzing information, constructing models, formulating arguments, designing artifacts, and generally processing ideas [Halasz 1988]. NoteCards has been used extensively within Xerox and in university environments and has been applied to a variety of tasks including document authoring, legal argumentation, development of instructional materials, and market analysis.

Data model and underlying metaphor. A NoteCard database consists of one or more notefiles containing interlinked notecards. Notecards must be organized into fileboxes within a notefile. A user may link their notecards into arbitrary, non-hierarchical network structures. The central metaphor is an electronic generalization of 3x5 paper notecards which are often used as an idea structuring tool.

Nodes. A node in this system is the notecard, which may contain an arbitrary amount of information embodied in text, graphics, or bit-mapped images. There are nine types of notecards differentiated, in part, by their contents, and users may define additional types. Notecards are also named.

Notefiles are stored as a file by the underlying operating system; fileboxes and notecards are indexed units within the file. The maximum size of a notecard, notefile, and NoteCard database is limited only by the underlying operating system, which also provides protection.

Links. In NoteCards an entire card may be a source anchor and there may be additional source anchors contained within. Source anchors within a card are icons. Destination anchors are

notecards. More than one "to" link may be associated with each source anchor. There are seven system-defined link types, and users may name links by attaching labels that characterize the nature of a relationship between two cards. "Back" links are provided, and link information is stored along with node contents.

Authoring. Authoring and browsing are combined into one system in NoteCards, and users may edit the contents of a notecard at any time. The user interface is mouse and menu based. Notecards are displayed in multiple, overlapping, scrollable Xerox Lisp windows. Source and destination anchors are specified, links are established, and notecards created and deleted through a series of cursor movements and function calls. Anchors are created at the time links are forged. NoteCards draws on a set of approximately 100 Lisp functions developed by its designers for most of its operations. Text and graphics may be imported, and facilities are present for printing the contents of one or more notecards. Versioning is not supported at the notecard level; however, whole notefiles are versioned when notefile compaction is requested.

Browsing. Source anchor icons are selected by positioning the mouse cursor and clicking. Destination notecards are displayed in a new window that remains open until explicitly closed. It is possible to show all links coming into and leaving a notecard, and the system provides a search facility that searches on contents and link type. Searching link types provides a limited structure search mechanism. The candidate names resulting from a search are placed in a newly created notecard, thus providing a limited view facility. Histories are provided only in the sense that accessed notecards remain displayed.

A graphical browser provides structural diagrams of linked notecards. The user may edit the underlying structure of this network by carrying out operations on nodes and edges. NoteCards is fully integrated into the Xerox Lisp programming environment and is extensible.

Usability and performance. NoteCards has been used successfully in a wide range of applications and much has been learned about hypertext system design by analyzing this system [Halasz 1988]. The most important drawbacks that we see relate to the NoteCards user interface. NoteCards does not have a front-end *per se*; it relies on the Xerox Lisp environment. Multiple, small, overlapping

windows containing potentially large amounts of information through which a user must scroll presents, at times, a difficult working environment.

References.

Halasz, F. 1988. Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems. *Communications of the ACM*, Vol. 31, No. 7, (July), 836-852.

Halasz, F., T. Moran, and R. Trigg. 1987. Notecards in a Nutshell. *Proceedings of the ACM Conference on Human Factors in Computing Systems, (CHI+GI)*, Toronto, (April).

Intermedia

Intermedia is a distributed hypertext system for workstations and personal computers. The system was developed by Norman Meyrowitz and a large team of researchers at Brown University's Institute for Research and Scholarship (IRIS) and was funded by grants from IBM and the Annenberg/CPB Project. Intermedia has been in development since 1985. It is presently available for the IBM PC/RT running a derivative of Berkeley Unix 4.3 and Apple's Macintosh and will likely be ported to other platforms. It supports color, high-resolution monitors and CD ROM.

Intermedia was expressly designed to support teaching and research in a university environment. It has been used in a variety of applications including teaching music theory, plant cell biology, and English literature, as well as in authoring, and collaboration. It also provides a framework for developing other object-oriented, direct manipulation editors and applications [Yankelovich *et. al.* 1988].

Data model and underlying metaphor. An Intermedia database consists of a set of arbitrarily interlinked documents containing information from a variety of applications. Documents may be organized into folders. The central metaphor is that of a desktop displaying documents in folders.

Nodes. A node in Intermedia is a document that may contain text, graphics, bit-mapped images, 3-D images, chronological timelines, music, and video. There is no limit to the size of a document or the number of documents in a database, except those imposed by the underlying operating system. Documents are typed and named. Intermedia supports shared, concurrent access to documents and link information and provides access control. Documents are stored as files by the underlying operating system.

Links. Source and destination anchors may range from a point within a document to the entire document and include text, graphics, bit-mapped images, or any other valid selection. Links may be named and typed by attaching attribute/value pairs called keywords. Multiple bi-directional "into" links may be attached to anchors. "Back" links are provided.

In Intermedia, anchor and link information is not stored within documents, but superimposed on them. Webs maintain the anchor and link information, allowing one or more users to work within

their own context undistracted by anchors and links created by others sharing the same documents. Data associated with anchors, links, attributes, and webs are stored as tuples in the Ingres relational database system.

Authoring. There is no mode boundary between author and browser in Intermedia; provided they have appropriate access rights, a user may directly manipulate the contents of a document at any time. There are several author applications available for editing text, graphics, bit-mapped images, etc. The act of specifying anchors and making links was modeled as closely as possible on the Smalltalk/Macintosh copy/paste paradigm. Anchors are specified at the time links are forged.

Much of the functionality required to perform editing operations is invoked using cursor selection from palettes -- sets of controls attached as a pane to a document window. Use of pull-down menus has been avoided where possible. Versioning has not, as yet, been implemented.

Browsing. Documents in Intermedia are displayed in overlapping, scrollable windows. Anchors are distinguished by icons and highlighting. An anchor is selected by pointing a mouse cursor and clicking a button; depending on the context, there exists three ways of specifying how to follow links that leave a selected source anchor. Destination documents are displayed in a new window.

Graphical overviews are provided that show the entire linked structure of a web or links that enter and leave a single document. Histories and indexes are not implemented, but Intermedia does provide filters, views, and a search facility based on the attribute/value pairs attached to links. Intermedia is not extensible.

Usability and performance. Intermedia is an appropriate system for teaching, research, and personal information management and presents a number of innovative design features. Its most serious drawbacks relate to problems of orientation; intricate webs over information are provided without adequate visualization mechanisms. The system is not currently available in the Hypertext Research Lab; therefore, it is difficult to comment further on usability and performance.

References.

Meyrowitz, N. 1986. Intermedia: The Architecture and Construction of an Object-Oriented Hypermedia System and Applications Framework. *Proceedings of the Conference on Object-Oriented Programming Systems and Languages*, (September), 186-201.

Yankelovich, N., B. Haan, N. Meyrowitz, and S. Drucker. 1988. Intermedia: The Concept and the Construction of a Seamless Information Environment. *Computer*, (January), 81-96.

Garrett, L. N., K. Smith, and N. Meyrowitz. 1986. Intermedia: Issues, Strategies, and Tactics in the Design of an Intermedia Document System. *Proceedings of the Conference on Computer Supported Cooperative Work*, Austin, Texas, (December), 163-174.

Augment

Augment is a distributed hypertext system for DEC mainframes. It has been under development since 1962 with funding provided by Standard Research Institute, Tymshare, and McDonnell Douglas Corporation. Augment was designed by Douglas Engelbart and has evolved out of his pioneering work with NLS (Online System). It was primarily a research vehicle but is now used in-house by McDonnell Douglas and sold commercially.

Augment was designed for augmenting human intellectual capabilities. It was targeted particularly toward knowledge workers and researchers engaged in planning, analyzing, and designing in complex problem domains [Engelbart 1984]. The system has been used extensively in team collaboration and has been used to produce and maintain system documentation.

Data model and underlying metaphor. Augment employs explicitly structured files with hierarchically organized nodes, called statements, that contain information. Statements are organized into structures by links; structures are organized into documents. Statements may be referenced individually, so an arbitrary graph structure is possible. The central Augment metaphor is an integrated environment that extends the capabilities of intellectual workers -- the "augmented knowledge workshop."

Nodes. A node in Augment is a statement that may contain text, graphics, a bit-mapped image, or other forms of useful data such as digitized speech or "electronic signatures." Statements may contain up to 2000 characters of text, and there is no limit to the number of statements that may exist in the database except that imposed by the operating system. Statements are named and may be typed through an associated property list. Augment provides shared, concurrent access to nodes and enforces protection. A document containing linked statements is mapped to a single file in the underlying operating system. For a more complete description of Augment's rather complex storage mechanism, see Kacmar *et. al.* [1988].

Links. Source anchors in Augment are a sequence of characters that specifically reference the destination node. Destination anchors may be any sequence of characters or a bit-mapped image. A single named, typed "into" link may be associated with each source anchor. "Back" links are

provided as well as indirect links that cause the destination anchor to be computed from an anchor in an intermediate statement. Links are stored in a document file along with statements.

Authoring. Author and browser are combined into one system in Augment. Specific details concerning Augment's user interface are lacking. Statements are displayed in multiple windows. Source anchors are specified by including an explicit textual reference to the destination in the text of a statement. Text and graphics may be imported, and a print facility is provided. Augment supports versioning and collateral display.

Browsing. In the Augment browser, multiple threads of navigation may be displayed in arbitrary rectangular windows. Source anchors are selected by mouse cursor. Specific details concerning the user interface to the browser are not available; however, Augment does provide a number of features to assist the user in navigating the database. A variety of indexes and an elaborate search mechanism is provided. In addition, Augment provides a number of filters that generate differing views over information in documents. Histories and graphical overviews are provided. Augment is not extensible.

Usability and performance. The Augment hypertext system appears useful for a wide range of personal and collaborative problem-solving applications. The system is not available in the Hypertext Research Lab; therefore, it is difficult to comment further on usability and performance.

References.

Engelbart, D. 1984. Authorship provisions in Augment. *Office Automation Conference Digest*, 465-472.

Engelbart, D. 1984. Collaboration support provisions in Augment. *Office Automation Conference Digest*, 51-58.

Augment: Applications and features. 1986. McDonnell Douglas Corporation.

Neptune / HAM

Neptune/HAM is a distributed hypertext system designed by Norman Delisle and Mayer Schwartz at Tektronix's Computer Research Laboratory. It is a research vehicle intended to provide database support for engineering environments, especially computer aided design (CAD), computer aided software engineering (CASE), and engineering information systems (EIS). Neptune is primarily a user interface, implemented in Smalltalk-80, and HAM (Hypertext Abstract Machine) is a general-purpose, transaction-based, multi-user server for a hypertext storage system. HAM runs under Berkeley Unix 4.3. In describing Neptune/HAM, we are generally referring to the functionality provided by HAM [Delisle and Schwartz 1986].

Tektronix has recently sold its CASE division, and future development plans for Neptune/HAM are not known.

Data model and underlying metaphor. The HAM storage model is an arbitrarily-linked, object-based graph structure. The five objects in the system are: graphs, contexts, nodes, links, and attributes. A graph is the highest level object and usually contains all of the contexts relating to a given topic. Contexts partition nodes within a graph. Nodes contain information. Arbitrarily forged, bi-directional links define relationships between nodes. Attribute/value pairs attached to contexts, nodes, or links give semantics to HAM objects and can represent application-specific properties of objects or contain information that further describes an object. There is no central metaphor to Neptune/HAM.

Nodes. Nodes contain information stored as arbitrary amounts of text or fixed-length binary blocks. There is no limit to the size of a node or to the number of nodes in a database. Protection may be specified by attaching an access control list to all five objects managed by HAM. Nodes are stored in a centralized file server that allows distributed access. Synchronization is handled by a transaction-based mechanism built into HAM.

HAM assigns each object in the system a unique object identifier when it is created. Any number of additional names may be applied to an object by attaching attribute/value pairs. Nodes may also be typed by attaching an attribute. Nodes are stored as Unix files.

Links. Links in Neptune/HAM provide a level of flexibility and sophistication not seen in many existing hypertext systems. Links are stored separately as objects. They link nodes to nodes and are bi-directional, but users may attach attribute/value pairs to links to make them whatever they like. For example, points may be specified as anchors by attaching offset attributes; spans may become anchors by attaching offsets and lengths. This flexibility allows "to", "into", and other kinds of links to be created easily.

Links are typed and named by attaching attribute/value pairs. An anchor may have multiple links. "Back" links are provided. Links are independent objects stored separate from nodes, and it is possible to attach access control lists to links, thereby restricting sharing.

Authoring. The author and browser are combined into one system in Neptune/HAM. Creating and deleting nodes, links, and anchors are performed through function calls to the HAM back-end. Specific details concerning the user interface to the authoring component are not available. Nodes are displayed in seven standard browsers. They provide graphical overviews, allow editing of node contents and attributes, and show the hierarchical structure of a document. A node difference browser provides collateral display -- two versions of the same node are displayed side-by-side with differences highlighted.

A major design feature of Neptune/HAM is support for versioning. HAM provides an automatic version history mechanism on nodes and also maintains version history trees at the context level. Nodes may be printed. Neptune/HAM's import and export capabilities are not known.

Browsing. A user may display the contents of a node in a variety of ways using the standard browsers. Specific details concerning the user interface to the browser are not available; however, Neptune/HAM does provide a number of features to assist the user in navigating the database. Arbitrary search and query is possible based on attributes attached to contexts, links, and nodes, and node contents may be searched for the occurrence of user-specified regular expressions. In addition, HAM provides an elaborate filtering mechanism that allows subsets of HAM objects to be extracted from large nodes. Contexts themselves provide a view facility. The system is extensible in that any system may be built on top of the HAM back-end.

Usability and performance. The Neptune/HAM hypertext system is useful for a range of CAD and CASE applications. This system is not available in the Hypertext Research Lab; therefore, it is difficult to comment further on usability and performance.

References.

Delisle, N., and M. Schwartz. 1986. Neptune: A Hypertext System for CAD Applications. *Proceedings of the ACM Sigmod Conference*, Washington D.C., 132-143.

Campbell, B., and J. Goodman. 1988. HAM: A General Purpose Hypertext Abstract Machine. *Communications of the ACM*, Vol. 31, No. 7, (July), 856-861.

HyperTIES

HyperTIES (Hyper The Interactive Encyclopedia System) is a hypertext system for personal computers. It was developed at the Human-Computer Interaction Lab at the University of Maryland under the direction of Ben Shneiderman. HyperTIES has been under development since 1983, first as a research vehicle and recently as a commercial product. The system is currently available for color IBM PCs; future plans include porting the system to Sun workstations, adding support for high-resolution graphics, and touchscreen selection. Version 3.0 is scheduled to be released soon. This analysis reports on HyperTIES - Version 2.2 running on an IBM PC/AT.

HyperTIES is intended for a variety of uses including instruction, on-line help, diagnostic problem solving, organization policy making, and museum displays. It has been used as a presentational system at photography exhibitions, in a University of Maryland Student Union kiosk, and in the National Museum of Natural History [HyperTIES Reference Manual 1987].

Data model and underlying metaphor. A HyperTIES database consists of a set of arbitrarily interlinked articles. A related set of articles is organized into an encyclopedia. While the resulting structure is commonly a graph, introductory articles may impose a hierarchical organization. The central metaphor is a richly interconnected set of related articles that comprehensively cover a particular topic.

Nodes. A node in HyperTIES is an article. An article may contain either text or a bit-mapped image. Textual articles may range in size from 0 to 10,500 characters and are composed of a main body of text, optional heading, and optional two-line definition. Picture articles contain a single, screen-sized, bit-mapped image. A maximum of 200 articles may exist in a given encyclopedia. Articles carry names and synonyms and are typed as either text or picture. HyperTIES does not support shared access, and the only protection is that provided by the underlying operating system.

An encyclopedia is a DOS directory. Articles are stored as DOS files, with text articles carrying .FIL extensions and pictures carrying a .PIC extension.

Links. Source anchors in HyperTIES are restricted to character strings. Destination anchors are entire articles. A single untyped, unnamed "to" link may be attached to a source anchor. "Back" links are provided in the author, but not in the browser. Link information is stored along with node contents.

Authoring. Author and browser are separate programs in HyperTIES and are invoked independently from the DOS command interpreter. The HyperTIES system is marketed as a complete package. The textual body of an article, its heading, and definition parts are displayed in paged, tiled windows. Functionality for editing, creating, and deleting articles is invoked by using keyboard arrow keys to position a highlight over operations appearing in a command pane, then selecting by pressing the "Return" key.

Source anchors are distinguished in the author by being bracketed by special symbols. The character string composing the source anchor is the name of an article or its synonym. The process of specifying anchors and creating links are two distinct steps in HyperTIES. Anchors without attached links are not distinguishable in the browser. The author only edits text; files containing bit-mapped images must be created by other graphics programs such as PC Paint. There is a limited print facility, and files may be imported.

HyperTIES does not support versioning or collateral display, and there is only limited facilities for annotating articles.

Browsing. Articles are displayed one at a time in tiled, paged windows within the browser. Source anchors appear in contrasting color to surrounding text. Source anchors are selected by using keyboard arrow keys to position a highlight over the anchor, then pressing the "Return" key. Links are traversed in two steps. First, the definition lines of the destination article appear in a window. If the user wishes to read the entire article, the "Return" key is pressed again, and the article's heading and text part are then accessed and displayed. Selectable options in a command pane allow the user to return to a previous article, page forward in the presently displayed article, or go to an index of articles.

A stack history is maintained but not displayed. HyperTIES does not provide graphical overviews, search facilities, or views, and is not extensible.

Usability and performance. HyperTIES is a minimal hypertext system useful for a range of presentational applications. Disorientation may occur at times, but providing an index and displaying only one thread of navigation through the database limits the problem. The encyclopedia model may be more applicable when implemented on large screen displays. For its intended purpose, HyperTIES performs well.

References.

Shneiderman, Ben. 1987. User Interface Design for the Hyperties Electronic Encyclopedia. *Hypertext '87 Papers*, (November), 189-194.

HyperTIES Reference Manual - Version 2.2. 1987. Cognetics Corporation, Princeton Junction, NJ.

Guide

Guide is a hypertext system for personal computers. The system was originally developed as a research vehicle at the University of Kent, England, by Peter Brown. It has been marketed by Owl International, Inc. since 1987.

Guide is a tool for creating electronic documents for a broad range of applications. It is advertised as "the first commercially available hypertext system for the PC's" [Guide Reference Manual 1988]. Guide is available for the Apple Macintosh and IBM PC under Microsoft Windows and supports both color and monochrome displays. The system is under development and may soon become commercially available for Sun workstations. This analysis reports on the IBM PC version of Guide.

Data model and underlying metaphor. Guide manages a database of guidelines interlinked through buttons. Guidelines are DOS files; a button is an anchor and its associated link. A Guide database is strongly hierarchical by virtue of its predominant linking mechanism, which implements stretchtext, but arbitrary graph structures may be created. There is no central metaphor to Guide other than it being a tool for electronic "desktop communication."

Nodes. A node in Guide is a guideline stored as a DOS file. It may contain text and bit-mapped images created by any of the Microsoft Windows graphics programs such as MS Windows Paint. The maximum size of a guideline and the maximum number of guidelines in a database are limited only by the DOS file system.

Guidelines are given DOS file names with a .GUI extension, but are not typed. The only protection is that provided by the DOS file system, and there is no support for shared, concurrent, or distributed access.

Links. Source anchors may be any character string or a bit-mapped image. Destination anchors may be text, bit-mapped images, or a guideline. A single untyped, unnamed link may be associated with a source anchor to create a button. There are three kinds of links in Guide. The replacement button is an anchor associated with a "stretch" link and is the predominant linking mechanism in Guide. Others include the reference button, which is an "into" link, and the note button, which is an

"include", "to" link. A fourth button, called an inquiry button, causes the system to enforce mutual exclusion on a group of replacement buttons. "Back" links are not provided. Link information is stored along with guideline contents.

Authoring. Author and browser are combined into one system in Guide. Source and destination anchors are selected from open windows using a mouse cursor, and link functions are invoked through pull-down menus. Anchors are specified at the time links are created. Text and graphics may be imported, and there is a facility for printing those portions of a node displayed in a window. The textual and graphic contents of a guideline may be modified at any time using a variety of editing functions. Several bit-mapped images may be combined to form a more complex, composite image with selectable components. Guidelines are saved by invoking a system function; however, deleting a guideline requires that the user exit Guide and invoke a DOS command to remove the file.

Browsing. Guidelines are displayed in overlapped, scrolled windows. Multiple threads of navigation through linked guidelines may be browsed. Source anchors are distinguished by a combination of fonts and cursor sensitivity. Replacement buttons appear in bold face type and the cursor changes into a "cross hair" sight when positioned over the anchor. A reference button appears in italics, and the cursor becomes a large arrow, and a note button is underscored, with the cursor changing into an asterisk. Image source anchors are distinguished by cursor sensitivity alone.

Anchors are selected by positioning the cursor and clicking the left mouse button. If a replacement link is followed, stretchtext is inserted. The cursor becomes a square when positioned over stretchtext. If a reference link is followed, it will go to some other reference in the same guideline or to a point in another guideline. In the latter case, another window is opened automatically to display the reference point. Following a note link causes the destination anchor to be displayed in a small window as long as the mouse button is pressed.

A stack history is maintained but not displayed. In order to return to the previous context after following a replacement link, the left mouse button is clicked when the cursor is positioned over stretchtext. The stretchtext then becomes folded under its replacement button. In order to return from a reference point, a backtrack icon on the window border is selected.

A limited search facility allows the textual contents of nodes to be searched. Guide does not provide a graphical overview and is not extensible.

Usability and performance. Guide is a minimal hypertext system useful for a limited range of personal information management applications. The user interface is poorly designed, and in many situations invoking functionality requires several confusing menu selections. It is often difficult to remain oriented while browsing guidelines. For example, there are two ways of returning to a previous context, selecting a backtrack icon and collapsing stretchtext. This requires the user to remember how they arrived at a particular location. Following a series of nested "stretch" links may also lead to disorientation.

Despite its drawbacks, Guide is remarkable in that it is one of the few existing hypertext systems to implement stretchtext.

References.

Guide Reference Manual. 1987. Owl International, Inc., Bellevue, WA.

HyperCard

HyperCard is a hypertext system for Apple's Macintosh personal computer. It has been under intense development at Apple since 1985. Bill Atkinson designed HyperCard. It was released in 1987 as a commercial product intended to make the Mac more accessible to users. Future releases of HyperCard will support CD ROM and high-resolution color graphics. This analysis reports on HyperCard - Version 1.1 running on a Mac II.

HyperCard is designed as a personal toolkit for managing information. It has been used in a variety of applications including computer aided instruction (CAI) and as a presentational system for museums and information booths [HyperCard User's Guide 1987].

Data model and underlying metaphor. A HyperCard database consists of stacks of cards arbitrarily interlinked by buttons. Stacks may also be used to group cards into hierarchies. The central HyperCard metaphor is the 3x5 paper notecard which is often used to manage information.

Nodes. A node in HyperCard is a card that may contain text, graphics, bit-mapped images, video, voice, and sound. There is no limit on the number of objects in a card, and the number of cards in a stack is restricted only by the underlying operating system. HyperCard provides password protection at the stack level. Cards are named internally by the system but are not typed. Shared access is not supported. A stack is stored as a file by the underlying operating system.

Links. Source anchors in HyperCard may be any object contained in a card, such as text, graphics, a bit-mapped image, etc. The destination anchor may be a card in any stack. A single named, untyped "to" link may be attached to a source anchor to create a selectable button. A script consisting of a HyperTalk program may be associated with a button so that following a link may also start a process. "Back" links are not provided. Link information is not embedded in the contents of a card but stored separately and overlaid.

Authoring. Author and browser are combined into one system in HyperCard, and a user may directly manipulate the contents of a card at any time. Cards are displayed in a single window in the center of the screen, and functionality is invoked through a mouse cursor and pull-down menus.

The HyperCard user interface has the concept of "space", so text, graphics, and buttons may be positioned anywhere on a card. In addition, cut and paste operations may be performed on bit-mapped images. Text and graphics may be imported, and cards may be printed. Versioning is not supported in HyperCard.

Browsing. In HyperCard, buttons may be displayed in a variety of ways, such as by name, highlight, or as an icon. They may also be transparent. Buttons are selected by mouse cursor. Other navigation options include automatic scan, go to the next or previous card in a stack, go to the first card in a stack, etc. HyperCard provides a search facility and a history, but does not implement filters or provide views. HyperCard is extensible through the HyperTalk programming language.

Usability and performance. HyperCard is currently a minimal hypertext system useful for a range of personal information management applications and as a presentational system. Displaying a single, small card is, at times, restrictive and may lead to orientation problems. However, the system's responsiveness, direct manipulation interface, accessibility, and general appeal make it appropriate for its intended purposes.

References.

HyperCard User's Manual. 1987. Apple Computer Corporation. Cupertino, CA.

Concordia / Document Examiner

Concordia/Document Examiner is a distributed documentation development and management environment available for the Symbolics 3600 series Lisp computers running the Genera 7.2 Operating System. They were developed at Symbolics under the direction of Janet Walker. Concordia is the documentation development component and has been used in-house at Symbolics since 1983. It was made available to consumers for the first time this year with the release of Symbolics' Genera 7.2 Operating System. Document Examiner is the delivery interface for on-line lookup and has been supplied to consumers as part of the operating system for several years. The products are presently under development. This analysis reports on Concordia Beta-test Version (Version 1 due in May 1988) and the Genera 7.2 Version of Document Examiner.

Concordia/Document Examiner is an integrated environment for professional technical communication, especially the writing, editing, illustration, design, production and maintenance of large documentation sets having very long life cycles, maintained by a team of writers [Walker 1988]. The system has been used at Symbolics for producing most of their recent system software documentation. The Concordia environment is strongly biased towards the production of linear, paper-based documents for publication. It is more accurate to view Concordia/Document Examiner as a documentation development system having hypertext system functionality rather than viewing it as a hypertext system *per se*.

Data model and underlying metaphor. A Concordia/Document Examiner database consists of structured records linked together to form a document. A record is the smallest unit of information accessible from Document Examiner and is the basic building block from which writers construct entire documents. The final organization of a document is a result of the structure imposed by the way records are linked. Since the system is designed for publishing books, the data model tends to be strongly hierarchical; however, arbitrary graph structures may also be formed. There is no central metaphor to Concordia/Document Examiner.

Nodes. A node in Concordia/Document Examiner is a record composed of a set of fields: type, name, content, one-liner, keyword, and notes. Records may be of two general types: concept and object. Concept records encompass text and illustrations and account for the majority of records in the database. Object records are reserved for documenting Lisp objects. Records are named, may carry a brief one-line description, and a list of keywords that appear within the record to facilitate searching. The contents field may contain text or imported graphic images, formatting markup, and links to other records.

The only limits placed on record size or number of records in a database are those imposed by the operating system. A database may be shared and accessed concurrently. The underlying operating system provides protection and handles concurrent updates. Records are stored as files with .SAB file type.

Links. Source anchors in Concordia/Document Examiner are the name of a record. Destination anchors are fields within a record. A single typed, unnamed link may be associated with a source anchor. "Back" links are provided. Information about the links present in a record is stored along with record contents.

Linking options in Concordia/Document Examiner are fairly complex and reflect the system's document production orientation. There are five link types: include, contents, cross-reference, topic and precis. All are "automatic include", "to" links that filter fields in the destination record. Include links retrieve the name and contents fields of the destination and are distinctive in that they define the hierarchical structure of a developing document by specifying table of content entries. The content link is similar to the include, except it retrieves only the contents field of a record. The precis, cross-reference, and topic links all retrieve the record name field automatically and make the name mouse selectable. The precis link also accesses the one-liner field and displays it below the record name; the cross-reference link causes the words "See the" to precede the record name. In each case, when the name field is selected in the browser, the name and contents field of the destination record are retrieved.

Authoring. Concordia is the authoring component of the system and is marketed separately from Document Examiner. The user interface presents three tiled, scrollable windows: a large pane for editing text and graphics, a pane displaying operations that may be invoked by selection with a mouse cursor, and a current records pane. Anchors and links are specified, and records created, deleted, and accessed through function calls. Anchors are specified at the time links are created. Linked anchors are preceded by distinguishing symbols. Text and graphics may be imported into a record, and documents may be printed. The print formatter assembles a document by following links in a depth-first manner.

Concordia offers an array of editing features, fonts, stylesheets, and a markup language. In addition, it provides search capabilities, graphical overviews, and a previewer which shows a scaled image of a document page. The editor is not WYSIWYG. Versioning is provided at two levels: versions of an entire document database are maintained as well as versions for each record within the database. Annotation may be added to a record in the notes field.

Browsing. Document Examiner is the browser component of the system. It is integrated into the Genera operating environment and may be conveniently invoked in any context to access information on-line. In the case of Symbolics' system documentation, the on-line version has the same contents as the printed manuals because both are produced from the same database.

The user interface for Document Examiner consists of four tiled, scrollable windows. The viewer pane is the largest and displays a single thread of navigation through record contents. Other panes include a candidate pane, a history pane, and command pane. As records are retrieved for display in the viewer pane, all "automatic include" links are followed, and the record is dynamically built. Cross-reference and topic anchors are enclosed in a box and are selectable by pointing with the mouse cursor and clicking a button. Retrieved records are then appended to what has already been displayed in the viewer. Table of contents entries, indexes, and the results of search operations (all record names) may be displayed and selected in the candidate pane. The history pane displays the names of records that have appeared in the viewer and may be saved for future reference or to direct printing of a subset of manual pages. This provides a limited view facility.

Graphical overviews are provided along with a number of other functions designed to assist navigation, such as the ability to display links to and from a record. In many cases, it is possible to execute lines of Lisp code that appear in the documentation, and facilities are provided for copying displayed documentation into a Zmacs buffer for other uses. Document Examiner is not extensible.

Usability and performance. Concordia/Document Examiner is appropriate for large documentation applications that require frequent updating, have long life cycles, and have reusable sections. People who work extensively on Symbolics computers generally have favorable comments about Document Examiner, and it appears to function well as an on-line delivery interface. Maintaining orientation may be a problem at times because new information is added to that already displayed in the scrollable window. After several retrieves, a user may scroll backwards through what appears to be a jumble of seamless, unrelated pieces of information. The graphical overviews are helpful.

To date only the technical writers within Symbolics have had extensive experience with Concordia, so there is less known about its usability. It appears to support its intended use well.

References.

Walker, J. 1988. Supporting Document Development with Concordia. *Computer*, (January), 48-59.

Concordia Documentation Set. 1987. Symbolics, Inc., Cambridge, MA.

Genera 7.2 On-Line Reference for Document Examiner. 1988. Symbolics, Inc., Cambridge, MA.

VisiDoc

VisiDoc is a distributed hypertext browser for on-line documentation on the Texas Instruments Explorer series Lisp computers. It became available in January of this year and was developed by Ken Bice. There is no authoring component to VisiDoc -- it is a presentational system for on-line manuals which employs some hypertext system functionality. The system is no longer under development, and plans for its future use are uncertain.

Data model and underlying metaphor. A VisiDoc database consists of manuals organized hierarchically under a table of contents. Each entry in the table of contents stretches to make sections more specific or collapses to hide detail. The central metaphor in VisiDoc is stretchtext; a manual is one piece of stretchtext.

Nodes. A node in VisiDoc is a section or chapter in a manual and may contain text or bit-mapped images. A manual may be viewed as a composite node that brings in other nodes to assemble a document. The only limits on node size or the number of nodes in a database are those imposed by the operating system. Access to nodes is read only, and shared nodes may be accessed concurrently. Nodes are not typed, but may carry a section or chapter name. A manual is a directory in the underlying operating system; sections and chapters are stored as files within this directory.

Links. Source anchors in VisiDoc are titles or keywords that reference specific sections or chapters of a manual. Destination anchors are entire nodes. A single unnamed, untyped "stretch" link is associated with a source anchor. Links may not be forged between sections or chapters across manuals. "Back" links are not provided, and link information is stored along with node contents.

Authoring. Not supported in VisiDoc.

Browsing. The VisiDoc user interface consists of tiled windows, and manual entries are displayed in a large, scrollable viewer pane. Source anchors are distinguished by being boxed and are mouse selectable. Selecting an anchor causes stretchtext to be inserted into the present context, thus providing more detailed information. Stretchtext may be collapsed in order to return to a previous, less detailed context. The system does not implement an index, filters, or views, but does provide

graphical overviews, a search facility, and a history that is displayed in a separate history pane. VisiDoc is not extensible.

Usability and performance. VisiDoc is designed specifically for on-line display of a small amount of highly hierarchical TI Explorer documentation, and its use cannot be extended easily to other applications. The major drawback to its user interface is that the user may become lost in a scrollable window filled with pieces of text at various levels of expansion. VisiDoc is remarkable in having stretchtext as its central metaphor.

References.

TI Explorer On-Line Reference for VisiDoc. 1988. Texas Instrument Corporation. Austin, TX.

Tabular Comparisons of Selected Hypertext Systems

The following tables provide a summary of some of the more important characteristics of each system reviewed. For many features, a tabular presentation does not capture the degree of sophistication or subtle variation that may exist in a given hypertext system's implementation. The reader is encouraged to turn to the more detailed descriptions provided earlier in this report for more information.

Table 1. General Characteristics of Selected Hypertext Systems	p. 45
Table 2. Node Characteristics of Selected Hypertext Systems	p. 46
Table 3. Link Characteristics of Selected Hypertext Systems	p. 47
Table 4. Authoring Characteristics of Selected Hypertext Systems	p. 48
Table 5. Browsing Characteristics of Selected Hypertext Systems	p. 49

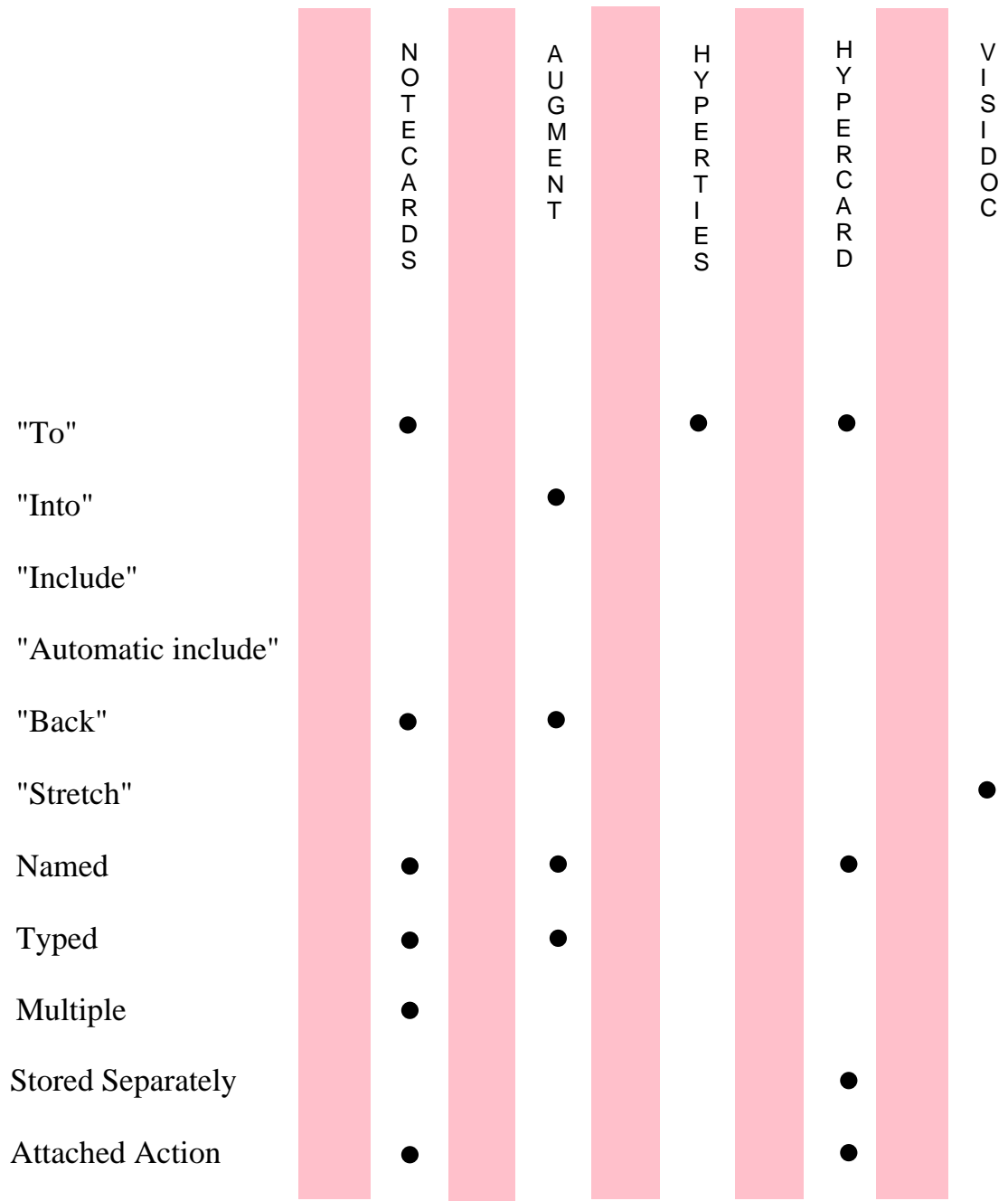
General Characteristics of Selected Hypertext Systems

	NOTE CARDS	AUGMENT	HYPER TIES	HYPER CARD	VISIDOC
Hypertext System	●	●	●	●	
Workstation Based	●				●
PC Based			●	●	
Networked		●			●
Research Vehicle	●	●	●		
Commercial Product	●	●	●	●	●
Under Development	●	●	●	●	
General Application	●	●		●	
Special Application			●		●

Node Characteristics of Selected Hypertext Systems

	NOTECARDS	AUGMENT	HYPERTIES	HYPERCARD	VISIDOC
Text	●	●	●	●	●
Graphics	●	●		●	●
Bit-mapped Image	●	●	●	●	
CD ROM, Other		●		●	
Named	●	●	●	●	●
Typed	●	●	●		
Shared Access		●			●
Concurrent Access		●			●
Distributed Access		●			●
Protected	●	●		●	●
Object Based				●	

Link Characteristics of Selected Hypertext Systems



Authoring Characteristics of Selected Hypertext Systems

	NOTE CARD S	AUG MENT	HYPER TIES	HYPER CARD	VISI DOC
Combined System	●	●		●	
Overlapped Windows	●	●			
Tiled Windows			●	●	
Collateral Display		●			
Space "Exists"				●	
Import	●	●	●	●	
Print Facilities	●	●	●	●	
Markup		●		●	
Annotation	●	●	●	●	
Attribute Attachment	●	●			
Versioning	●	●			

Browsing Characteristics of Selected Hypertext Systems

	NOTE CARDS	AUGMENT	HYPER TIES	HYPER CARD	VISI DOC
Tiled Windows	•	•		•	•
Scrolling	•	•			•
Multiple Threads	•	•			
Mouse Selection	•	•		•	•
Graphical Browser	•	•			•
Search	•	•		•	•
Index		•	•		
History		•		•	•
Views	•	•			•
Extensible	•			•	

Literature Cited

- Akscyn, R. M., D. L. McCracken, and E. A. Yoder. 1988. KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations. *Communications of the ACM*, Vol. 31, No. 7, (July), 820-835.
- Bush, V. 1945. As we may think. *Atlantic Monthly*, Vol 176, (July), 101-108.
- Campbell, B., and J. Goodman. 1988. HAM: A General Purpose Hypertext Abstract Machine. *Communications of the ACM*, Vol. 31, No. 7, (July), 856-861.
- Delisle, N., and M. Schwartz. 1986. Neptune: A Hypertext System for CAD Applications. *Proceedings of the ACM Sigmod Conference*, Washington D.C., 132-143.
- Engelbart, D. 1962. Augmenting the human intellect: A conceptual framework. Stanford Research Institute, Menlo Park, CA.
- Engelbart, D., R. W. Watson, and J. C. Norton. 1973. The augmented knowledge workshop. *Proceedings of the National Computer Conference*, New York, NY, 9-21.
- Engelbart, D. 1984. Authorship provisions in Augment. *Office Automation Conference Digest*, 465-472.
- Gregory, R., 1983. Xanadu: Hypertext from the future. *Dr. Dobb's Journal*, No. 75, (January), 28-35.
- Guide Reference Manual*. 1987. Owl International, Inc., Bellevue, WA.
- Halasz, F. 1988. Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems. *Communications of the ACM*, Vol. 31, No. 7, (July), 836-852.
- HyperCard User's Manual*. 1987. Apple Computer Corporation. Cupertino, CA.
- HyperTIES Reference Manual - Version 2.2*. 1987. Cognetics Corporation, Princeton Junction, NJ.
- KMS Reference Manual*. 1987. Knowledge Systems, Inc., Murrysville, PA.

Kacmar, C. J., J. Leggett, J. L. Schnase, and C. Boyle. 1988. Data management facilities of existing hypertext system (in press).

Nelson, T. H. 1965. A file structure for the complex, the changing and the indeterminate. *Proceedings of the ACM 20th National Conference*.

Nelson, T. H. 1967a. Hypertext note 1: Brief words on the hypertext. Personal communication.

Nelson, T. H. 1967b. Hypertext note 6: Presentation theaters for interactive media. Personal communication.

Nelson, T. H. 1967c. Hypertext note 8: Stretchtext. Personal communication.

Walker, J. 1988. Supporting Document Development with Concordia. *Computer*, (January), 48-59.