The Knowledge Engineering Review

Expert systems and hypertext

ROY RADA AND* JUDITH BARLOW**

**Department of Computer Science, University of Liverpool, Liverpool L69 3BX, England*
***US WEST Advanced Technologies, Englewood, Colorado 80111, USA*

Abstract

Expert systems and hypertext are technologies which are coming together. This paper reviews the strengths and weaknesses of expert systems and hypertext. Then the complementarity of the two is both argued and illustrated. Translating expertise from documents into expert systems is difficult, but hypertext systems exploit the information in documents. Expert systems don't explain their decisions as well as some people would like, but hypertext is basically designed to provide explanations. On the other hand, people have difficulty deciding what links to follow in hypertext, but an expert system is designed to help people make decisions. When a hypertext reader is confused as to what step to follow next, an expert system might review the path taken thus far and suggest the appropriate next step. An annotated bibliography of the principal literature is provided.

1 Introduction

Expert systems and *hypertext* are two major computer technologies that hold great promise. The hardware movement from mainframes to networks of workstations supports this promise, as does the progress in cognitive modelling and human-computer interface development. Furthermore, expert systems and hypertext are complementary. As expert systems explain their decisions to users, the rules on which the expert system is based often seem inadequate (Clancey, 1983). Hypertext can be integrated with the expert system rules so that textbook information can be offered to the user when the knowledge behind the rule is questioned. From the other side, hypertext systems need improvement. The network of connections in hypertext may confuse the reader. Hypertext systems can be extended by including expert system techniques within them that help guide the user to relevant information (Belkin et al. 1987). We believe that the combination of expert and hypertext (here called *expertext*) systems offers great promise.

Expert systems are designed to solve problems and hypertext systems to convey information. These two objectives are flip sides of the same coin. Hypertext or text is meaningful to the reader to the extent that it addresses a problem which the reader wants solved. The first dictate of communication is that the communicator should know the needs of his audience and forward a message that responds to those needs. Likewise, an expert system is communicating someone's problem solving expertise to someone who lacks that expertise. Solving problems with an expert system can thus be seen as communication.

In this section, we present a framework from which the developments in hypertext and expert systems can be viewed and also characterize the salient problems with both systems. That these problems are complementary is the theme of the paper. Next are sections which detail the history and current challenges for expert systems and hypertext. Section 4 gives examples of how hypertext features can be added to expert systems and how expert system features can be added to hypertext.

1.a Framework

Underlying the developments in hyptertext is a cognitive theory which notes that people are skilled at reading and writing linear prose but that deep in their minds the representation is more complex.

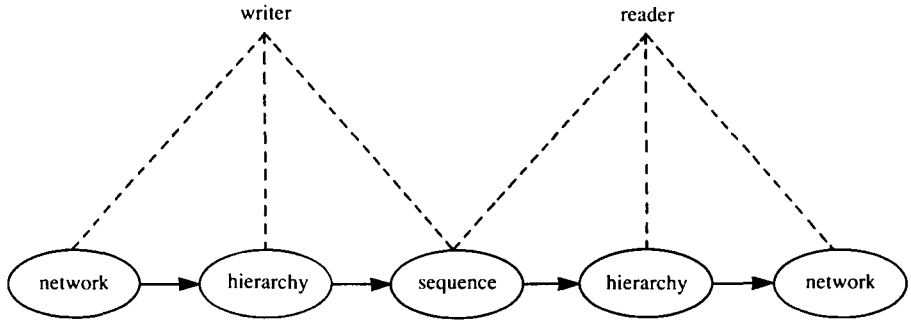

Figure 1 The representational forms through which a writer moves are a mirror image of those through which the reader moves

In one simplified interpretation, writers go from a network of ideas to text and readers translate text into a network of ideas. The intermediate stage between the network and text is a hierarchy (see Figure 1) (Smith et al. 1986). A hypertext system allows the author to directly express his network of ideas and presents to the reader this network of ideas. The intermediate steps of hierarchy and sequence can be sidestepped. Thus the writer and reader come closer together.

Expert systems are built by knowledge engineers and experts who together translate expertise into a knowledge base plus inferencing mechanism (Hayes-Roth, Waterman and Lenat, 1983). Users access the expert system and present it with characteristics of a problem which the expert would normally solve. The expert system then solves the problem. While this expert system process (see Figure 2) may bring the expert closer to the user than a document prepared by the expert for the user would, the expert system basically affects a kind of communication between expert and user. Since a hypertext system affects a communication between a writer and reader, the parallels between a hypertext and expert system might allow insights as to how to improve the communication powers or persuasiveness of expert systems.

1.b Problems

Some of the most famous expert systems have not gone beyond the development stage because they couldn't be easily integrated in the work place—they didn't communicate smoothly with the user. MYCIN, while expert within its domains of bacteremia and meningitis, does not cover enough of the information which physicians routinely need to manage for the physician to be willing to spend the necessary time to access MYCIN when a bacteremia or meningitis question arises. The developers of MYCIN are now focusing their attention on the development of user-friendly or hypertext information systems that incorporate expert systems but primarily handle the day-to-day work flow of a physician in a hospital ward (Shortliffe et al.).

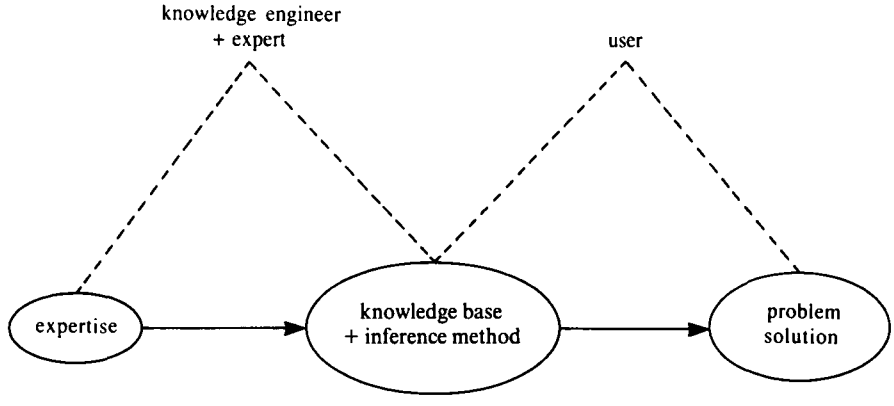

Figure 2 An expert system contains a knowledge base and inference method which a knowledge engineer and expert have built to solve problems of users

INTERNIST is an expert system that can handle a wide range of internal medicine problems. Over 4,000 signs and symptoms have been connected with over 1,000 diseases for diagnosis and treatment purposes (Miller and Myers, 1982). Unfortunately, the user must know how to phrase the patient's problem in the exact terms that the developers of the expert system chose—this discourages the user. A natural language interface could translate the user's phrasing into that of the expert system, but such a translator does not exist and alternatives are being pursued which take advantage of hypertext strategies. For instance, INTERNIST has been converted into a teaching tool which allows the user to browse through the knowledge in INTERNIST in a hypertext fashion. Thus the options of what to pursue at each point are made clear, and the user doesn't have to guess at the phraseology of one expert. Other medical expert-information systems are following the hypertext paradigm (Timpka, 1986).

When a document is delivered on paper, the reader proceeds by turning pages, but in a hypertext system the reader *browses* across networks. The reader is thus allowed to see the rich connectivity of ideas which the writer has in mind. Also the reader can pursue different perspectives on the document depending on which links are activated. The enthusiasm about hypertext is, however, dampened by the experience that readers tend to get lost when they are presented with many links from each frame of text (Raskin, 1987). One way to reduce the confusion seems to be to constrain the number of node and link types. Another strategy is to place more sophisticated help or guide facilities in the links or nodes. One of the greatest challenges to the development of successful hypertext systems concerns the insertion of expert procedural guides to browsing (Halasz, 1988).

This paper next presents a view on expert systems and details some of their common features in a way that facilitates comparison with hypertext. Hypertext, in turn, is defined and compared with expert systems. The relative strengths and weaknesses of the two technologies are shown to complement one another in such a way that the new technology expertext arises.

2 View on expert systems

2.a History

The goal of an expert system is to capture the expertise of a human expert in such a way that it is accessible and useful to a non-expert. In the 1970s, Bruce Buchanan and Edward Shortliffe developed MYCIN, an expert system which embodies expertise related to the diagnosis of bacteremia and meningitis. The fact that the domain knowledge is coded and stored separately from the mechanism which searches through it is an important feature that distinguishes MYCIN from traditional, procedural computer programs (Waterman, 1985). In MYCIN, domain knowledge related to medical diagnosis is expressed in the knowledge base as "if-then" rules. The inference mechanism which searches through the rule base is separate and distinct from the knowledge base. Meta rules, which are rules related to rule management rather than domain knowledge, are used to fine-tune how the knowledge base is traversed (Hayes-Roth, Waterman and Lenat, 1983).

The MYCIN project led to the development of tools which are portable to other domains. The essential non-medical framework of MYCIN has been abstracted to create the general purpose tool called EMYCIN (Buchanan and Shortliffe, 1984). There are many expert system shell packages on the market today which are modern extensions of EMYCIN including Personal Consultant from Texas Instruments Corporation and M1 from Teknowledge Corporation.

Alternative knowledge representation strategies such as logic programming, frames, and object-oriented designs also evolved during the 1960s. Prolog, a first-order implementation of predicate logic, has been adopted as the official language of the Japanese 5th Generation project (Gallaire, 1985). Object-oriented and frame-based structures are available in most commercial expert system development environments. Although the knowledge in each of these paradigms is represented differently than it is in MYCIN, many of the inference strategies are similar. In each of these expert system designs, an explanation facility can be easily provided by showing the path that the inference mechanism took through the knowledge base to arrive at a conclusion. Further, midrun explana-

tions as to why a particular piece of information is needed can be provided by showing the current state of inferencing and possible results that might be obtained if the user responds to the query.

Although many textbooks differentiate expert systems on the basis of how the knowledge base is structured, the current trend in expert system software appears to be one of incorporating numerous knowledge representation strategies in unified programming environments. Early implementations of large expert system development tools such as ART (Automated Reasoning Tool) from Inference Corporation, KEE (Knowledge Engineering Environment) from IntelliCorp, Knowledge Craft from Carnegie Group, and LOOPS from Xerox Corporation relied heavily on rules and frames for knowledge representation. Current versions of these systems have been expanded to allow for the incorporation of other knowledge representation paradigms including Prolog code, object-oriented structures, LISP procedures and procedural programs such as FORTRAN or C (Benchimol, Levine and Pomerol, 1987).

Knowledge Pro, a rule-based expert system development tool by Bill and Linda Thomson of Knowledge Garden, includes a hypertext explanation facility. When the user is queried to provide a piece of information, the prompt is presented in a hypertext format. If the user understands the query, she can proceed. If the user is unclear as to what information is required to answer the question or does not understand, then she can select the highlighted terms in the query and context-sensitive explanations are provided.

2.b Basic concepts

The major components of an expert system are knowledge or *expertise* and the facilities for applying that knowledge to problems (Hayes-Roth, Waterman and Lenat, 1983). This knowledge is often provided by an expert and herein lies one of the major barriers to expert system development, namely, the transfer of expertise from a human into a computer program (Rada, 1985). While many tools have been developed which either facilitate the process of computer knowledge acquisition or even completely automate it, none has proven able to consistently crack the shell which surrounds the nut (Forsyth and Rada, 1986). Neither the experts in an application area nor cognitive scientists have a sufficiently deep understanding of how experts represent and manipulate their knowledge to allow easy translation into an expert system.

The facilities which operate on the expertise are a reasoning tool for *solving problems* and an *explanation* tool for helping the user understand how the knowledge has been applied. Given that the knowledge is represented as "if-then" rules, the problem solver generally uses some combination of forward chaining and backward chaining. One common difficulty is that many rules fire and each rule may prompt the user for information which takes the user time and energy to provide (Buchanan and Shortliffe, 1984). In part, to compensate for the drain on the user's resources which repeated questioning entails and, in part, to increase the credibility of the conclusion, an expert system is expected to *explain* its behaviour. Often this explanation is provided simply by showing the user the rules and the sequence in which they were invoked. Mid-run explanations are often provided by showing a translation of the current goals and the rule currently being traced. The knowledge base rules are often obscure to all but the writer of the rules. Much research has gone into representing world knowledge in such a way that it clearly explains the expert's decision process to the non-expert.

2.c Strengths and weaknesses

A major strength of expert systems is that within a narrow domain they are able to capture much of the procedural expertise that a human expert uses to make decisions. Although expert systems often fail to perform at the same level of competence as the human expert, they can effectively serve as intelligent advisers for non-expert users who have some knowledge of the domain. An expert system can guide a user with limited knowledge in making decisions which are consistent with those that a human expert would make.

Weaknesses of expert systems include their inability to capture some of the important nuances of expert decision making (Alty, 1987). They may offer accurate advice for general or typical problems, but may be unable to deal with unusual or exceptional cases. Even when the advice offered is appropriate and accurate, the steps taken to arrive at a conclusion are not the same ones that a human expert would use. This makes it difficult for an expert system to explain its behaviour in a way that is useful to the human user (Stenton, 1987).

There have been a number of attempts to add intelligent explanation facilities to expert systems. While it is relatively easy to show the user a trace of the steps that the expert system took to arrive at a particular conclusion, the explanation offered is often unsatisfactory. Because the expert system processes information and arrives at conclusions in a different way from a human, the reasoning used is often unclear and appears awkward to the user. Much research continues in this area (Slatter, 1987).

An advantage of expert systems over traditional, procedural programs is their ability to offer mid-run explanations. Any time the expert system queries the user to provide a new piece of information, the user can ask for an explanation as to why the system requires further information. It is easy to explain why the expert system posed a specific query by showing the user the current status of the inference mechanism; however, the same problems apply as when offering the user an understandable explanation of how the expert system arrived at a conclusion (Carroll and Aaronson, 1988). Because the expert system searches its knowledge in a different way than a human expert would, the information provided is often unclear and confusing.

3 View on hypertext

3.a History

Vannevar Bush wrote a landmark paper in the *Atlantic Monthly* in 1945 which envisioned the creation of a "memex" or *mem*ory *ex*tender system (Bush, 1945). The memex system connected computer-like machines to large microfiche files. Users browsed the microfiche by issuing simple commands at a terminal. Networks of such systems created a global information system.

In the 1960s Engelbart and colleagues at Stanford University developed workstations for the creation and browsing of documents (Engelbart and English, 1968). The mouse was developed to make the information on the screen easier to navigate. Around the same time an on-line system for the retrieval of bibliographic citations was developed at the National Library of Medicine. That system now contains millions of citations which are indexed by terms from an indexing language of more than ten thousand concepts which the user can browse (Mili and Rada, 1988).

The new, notecard technologies support the writer in creating the "notes on cards" which correspond to the first stage in document writing. An early, commercial notecard technology was developed by Xerox. The product, called NoteCards, supports sophisticated editing and browsing on the Xerox workstations (Halasz, Moran, and Trigg, 1987).

3.b Model

Kintsch notes that humans encode text into an episode memory (Walker and Kintsch, 1985). In the writing process people refer to a rich collection of knowledge structures and processes in the course of transforming their ideas into text (Hayes et al. 1987). The traditional paper document does not optimally support the cognitive processes which people use in reading and writing. While paper copies of text will remain vitally important, screens displaying high-resolution graphics in multiple windows and distributed databases supplying large reservoirs of information mean that paper is outdated for some applications.

A new method of delivering text is hypertext and contains three major parts:

1 a database of text,

2 a network which connects the text components, and
3 tools for creating and browsing this combination of text and network.

The *text database* has documents in electronic form. The term text is misleading in that in practice hypertext systems have graphics. A more accurate term is hyperdocument. As documents are principal conveyors of culture and thus also expertise, the text database is typically provided by experts. The links and nodes which form the network of hypertext may be broadly construed as forming a conceptual net.

A number of general models of hypertext have been recently proposed. In one approach, Petri nets are fundamental to a formal view of hypertext (Stotts and Furuta, 1988). The hypertext is viewed as a sextuple $< N, C, W, B, M_l, M_d >$ in which

N is a Petri net,
C is a set of document contents,
W is a set of windows,
B is a set of buttons,
M_l is a logical projection for the document, and
M_d is a display projection for the document.

Garg (Garg, 1988) defined hypertext as a set of domain and information objects with predicates and attributes imposed on these objects. Garg particularly addresses the role of hypertext in representing the different versions of a document.

Object-oriented modelling offers an opportunity to unify approaches taken with hypertext and expert systems (Zaniolo et al. 1986). Object-oriented design requires that objects communicate with other objects via messages and that objects belong to hierarchies of objects. In expertext the objects hold text, and the messages or links between objects embody knowledge which can be used for problem solving. A link is an object which may send a message to another object which represents a document fragment. The object representing a document fragment belongs to a hierarchy of document fragments.

The various abstractions on hypertext have yet to provide a widely accepted model of the useful types of conceptual networks. In one view, the conceptual net in hypertext takes two principal forms: the independent one (Collier, 1987) and the embedded one. In the "independent" conceptual net the nodes of the net are tagged with concepts represented by terms. The "embedded" net may have a document chunk as a node. In the "independent" case each node of the conceptual net points to units of the document (see Figure 3), but the edges between nodes can be traversed without necessarily visiting the document units. In traversing an "embedded" conceptual net, the user may have to visit a large block of text (see Figure 4). In some of the better known hypertext systems, like NoteCards, the conceptual net is embedded within the document.

3.c Systems

HyperCard is an Apple product which follows in the footsteps of NoteCards. HyperCard is delivered automatically with every new purchase of a MacIntosh microcomputer from Apple Corporation. HyperCard takes advantage of the user-friendly interface of the MacIntosh and also incorporates most of the features of MacWrite and MacPaint (Goodman, 1987). Thus the average user has simple tools for text and graphics. Portions of a document are created on cards where each card fills the screen. "Buttons" may be inserted in cards and provide connections to other cards.

The creators of the hypertext software Guide have been particularly concerned about the "getting lost" phenomenon. Guide began as a research project at the University of Kent in Canterbury, England in 1982, but subsequently Office Workstations Limited translated key features of the original system so that it worked on microcomputers, including the MacIntosh and IBM-PC (Brown, 1987). The aim of the Guide developers was to display documents on computer screens in as ideal a way as feasible. The two principal buttons in Guide are the replacement and note buttons:

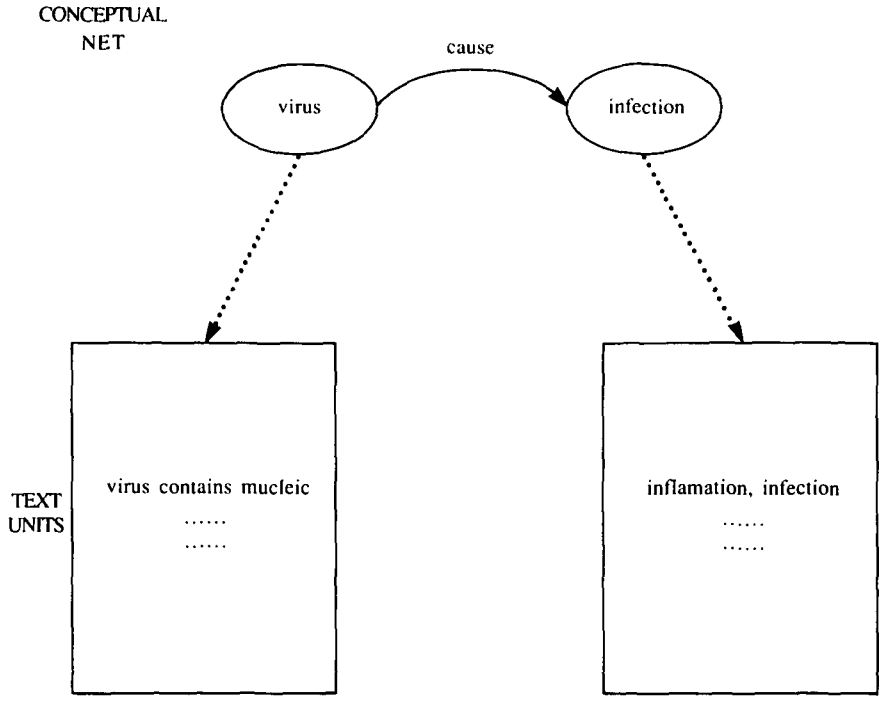

Figure 3 Sketch of conceptual net a level above the document itself

- *Replacement* buttons implement embedded menus (Koved and Shneiderman, 1986) which when selected with the mouse by the reader lead to an in-line replacement of the material linked with that button. The intent is that the replacement material will expand on material around the button.
- *Note* buttons are an extension of replacement buttons but display the additional material to which the note button points in a separate window of a split screen and keep it there only as long as the reader has the mouse activated over the note button.

These two buttons are designed to minimize the likelihood that the reader will feel lost. Both buttons keep the original material surrounding the button on the screen after the button is activated.

TIES allows novice users to explore information resources in an easy and appealing manner (Shneiderman and Morariu, 1986). Readers use arrow keys to move a light bar onto topics that interest them and a brief definition appears at the bottom of the screen. The users may continue reading or ask for details about the selected topic. An article about a topic may be one or more screens long. As users traverse articles, TIES keeps the path and allows reversal. Users can also select articles from an index.

Many documents are now typeset on the computer. The mark-up languages which are embedded

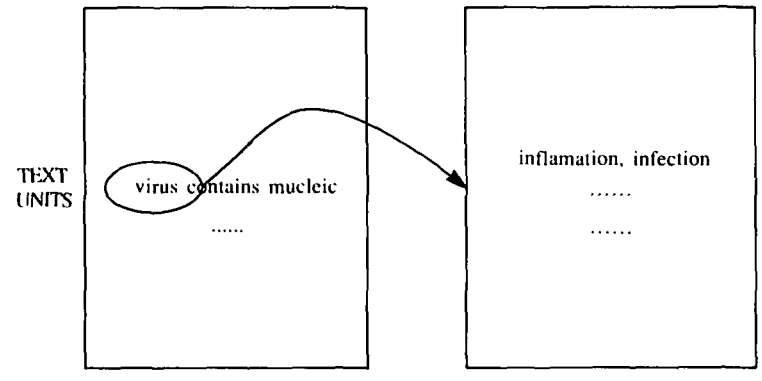

Figure 4 Sketch of conceptual net "embedded" within the document

in those documents can be used to create a simple hypertext. The Prototype Electronic Encyclopedia took an encyclopedia with formatting commands and produced a hyper-encyclopedia system (Weyer and Borning, 1985). The Thumb system took existing documents and made them appear as hypertext to the user (Price, 1982). The text was represented by a passage tree and a passage-dependent network. The passage tree was inferred from the source document formatting commands and thus captured, among other things, the hierarchical outline of the document. An expert in the domain of the document read the document and interacted with Thumb in the course of building the passage-dependent network which included links which couldn't be inferred from the formatting commands in the document.

KMS is a commercial hyperdocument system for networks of heterogeneous workstations which is a successor to the ZOG system developed at Carnegie Mellon University from 1972 to 1985 (Akscyn, McCracken, and Yoder, 1988). (In this section ZOG will, for simplicity of presentation, be considered an early version of KMS). The 1972 version of KMS provided a uniform menu-select interface to various computer programs. By 1983 a version of KMS was launched with the USS Carl Vinson, a nuclear-powered aircraft carrier. This version supported an interactive management system for analyzing and tracking complex tasks and an interface to an expert system. KMS is based on frames where each frame incorporates units of text and pointers to other frames (McCracken and Akscyn, 1984). The database consists of interlinked, screen-sized workspaces. While any kind of link can be used, KMS particularly supports hierarchical links. Users interact with the database by navigating from frame to frame, manipulating the contents of frames, creating new frames, and invoking frames. KMSs quickness depends on the graphics performance of the window system and the speed of the storage device, but response times under one second are typical. KMS provides to a community a single, logical database, physically distributed across multiple workstations and file servers on a network. The author/reader transition is automatic, and any changes to a frame are saved as soon as the frame is exited. Tools also exist for inheriting characteristics of one frame to another and for importing material from other sources, such as text files, into frames.

The Hypertext Abstract Machine (HAM) from Tektronix is a general-purpose hypertext storage system (Campbell and Goodman, 1988). HAM can be used as a base engine for other hypertext systems. While most hypertext systems emphasize the application and user interface layers, HAM addresses the storage model. HAM stores its database in a centralized file server and doesn't interfere with the other distributed file system functions of a network. The HAM storage model is based on five objects: graphs, contexts, nodes, links, and attributes. A graph is the highest-level object which, in turn, contains contexts. Each context has one parent context and zero or more child contexts. One sees here the hierarchical structure which is fundamental to HAM (and also KMS). Contexts contain nodes and links. Attributes can be attached to contexts, nodes, and links. HAM provides filtering mechanisms that allow predicates on contexts, nodes, links, and attributes to specify what objects from the database should be viewed. A version history is maintained such that earlier versions of an object can be viewed. The version history for a HAM object is updated each time the object is modified.

KMS is a commercial hyperdocument system for networks of workstations which has object-oriented features and which supports an expert system (Akscyn, McCracken, and Yoder, 1988). The Hypertext Abstract Machine from Tektronix is a general-purpose hypertext storage system with object-oriented features that support intelligent browsing (Campbell and Goodman, 1988).

4 Complementarity illustrated

While some of the inefficiencies of expert systems are overcome in hypertext systems, some of the weaknesses in hypertext systems are overcome by expert systems. This complementarity can be demonstrated by expanding the functionality of expert systems through incorporation of common hypertext features as well as augmenting hypertext systems by including expert guides and controls. We present examples of both these approaches to expertext.

4.a Integrating hypertext features in a rule-based expert system

Functional weaknesses in rule-based expert systems include the lack of consistently efficient inference strategies, inability to mimic the expert in explaining how and why a particular course of action should be taken, lack of intelligent help facilities, and inability to intelligently browse the knowledge base. Each of these weaknesses can be addressed by integrating hypertext features within the expert system environment. The addition of hypertext features to an expert system is illustrated next. In this illustration, an expert system shell called Personal Consultant Plus and a hypertext tool called Hype are used.

Personal Consultant Plus is an enhanced EMYCIN expert system shell from Texas Instruments Corporation. In addition to "if-then" rules, Personal Consultant Plus knowledge bases may include some frame-like features, textual and graphical help, mathematical and LISP functions, procedures, active values, operating systems calls, links to other programs, interfaces to other programming languages and more. Through the course of a consultation, the user can request an explanation as to WHY a particular question is asked as well as HELP in responding to a specific query. At the end of a consultation, the user can ask the expert system to "explain" HOW it arrived at a conclusion. The "explanation" provided is simply a translation and listing of the rules that were fired in sequence to arrive at the conclusion. This explanation, although presented in English, may be confusing and misleading for reasons discussed in section 2 of this paper.

Hypertext features can be added to Personal Consultant Plus through internal links to Hype, a small, public domain hypertext tool from Knowledge Garden. Hype is used to link all of the files related to a particular expert system including the development environment, knowledge base, help files, and meta level inference mechanisms. Calls to Hype are used to augment on line help, override the system explanation facility, and provide a mechanism for browsing the knowledge base.

Consider a small textbook expert system which helps a bank teller decide whether or not to cash a customer's cheque. Using Personal Consultant Plus, this can be accomplished most efficiently with a single goal and only one rule. The goal, which initiates backward chaining, is to offer appropriate advice to the bank teller. In order to satisfy that goal, the first rule which satisfies that goal is selected. In this case inferencing is greatly simplified as our expert system has only one rule, with the antecedent describing all of the criteria needed to make a decision to cash the customer's cheque:

```
IF:
   <customer-has-valid-id AND
   ((account-at-the-bank AND
    (sufficient-balance OR overdraft-privilege))
   OR
   (cheque-drawn-on-the-bank AND
    (sufficient-funds OR overdraft-protection)))>
THEN:
   advise teller to cash the customer's cheque.
```

A second rule might be formulated whose antecedent describes when to refuse to cash a customer's cheque; however, it is more efficient to set a default value on the goal of refusing to cash the cheque.

During a consultation the inference engine begins a trace on the first parameter in the rule's antecedent and the user is asked whether the customer has a valid id. If the user calls upon the explanation facility to explain why this information is needed, she will be informed that it is needed in order to decide whether to cash the customer's cheque. If the customer has a valid id, then the consultation continues with a trace on the second parameter, and the user is asked whether the customer has an account at the bank. If the user again calls upon the explanation facility to explain why this information is needed, she is again informed that it is needed to decide whether to cash the customer's cheque. The consultation continues and the user is asked either whether there is sufficient balance to cover the cheque or whether the cheque is drawn on the bank, depending on the response to the pre-

vious question. If the user calls upon the explanation facility for a third time, she will get the same response as the first and second time. Although the expert system can explain why it needs a specific piece of information, it exhibits an annoying lack of depth.

If, at the end of a consultation, the user asks HOW the expert system arrived at a conclusion, an English language translation of a rule will be displayed regardless of the advice offered. In the case where the antecedent conditions of the rule are satisfied, the user is informed that the decision to cash the customers cheque was made because there was either:

1 a valid id,
2 an account at the bank and sufficient funds,
3 an account at the bank and overdraft protection,
4 a cheque drawn on the bank and sufficient funds, or
5 a cheque drawn on the bank and overdraft protection.

When the advice is to refuse to cash the cheque, the explanation provided is that the set of antecedent conditions is not true. The precise setting of parameters which causes the success or failure of a rule is not available to the user. If, instead of the HOW facility provided with Personal Consultant, hypertext is used to browse text files linked to parameter settings, the user can discover precisely how and when a particular conclusion was reached and what conditions would need to change in order to arrive at alternative advice—perhaps precisely what is needed from the customer in order to allow the teller to cash the cheque.

An alternative approach would be to include a set of simple conjunctive rules which each have the same consequence.

```
IF:
   <customer-has-valid-id AND
   account-at-the-bank AND
   sufficient-balance>
THEN:
   advise teller to cash the customer's cheque.

IF:
   <customer-has-valid id AND
   account-at-the-bank AND
   overdraft-privilege>
THEN:
   advise teller to cash the customer's cheque.

IF:
   <customer-has-valid-id AND
   cheque-drawn-on-the-bank AND
   sufficient-funds>
THEN:
   advise teller to cash the customer's cheque.

IF:
   <customer-has-valid-id AND
   cheque-drawn-on-the-bank AND
   overdraft-protection>
THEN:
   advise teller to cash the customer's cheque.
```

This format can provide more meaningful explanations without adding hypertext features. The price, unfortunately, is a large and redundant knowledge base leading to inefficiencies in inferenc-

ing. The parameter for valid id is now repeated four times and may be traced as many times during inferencing. For a reasonably sized expert system, this approach may make resolution-unification unacceptably slow.

Hypertext approaches could make the explanation facility more useful. Suppose a query to explain why a particular piece of information is needed instead calls the hypertext system to bring up text linked to each of the parameters in the "if" portion of the single rule example presented earlier. If users ask why with reference to valid id, they can browse text about what constitutes a valid id as well as the consequences of different responses to this query. Users receive a different response if they call the explanation facility to ask why the customer's current account balance is necessary. Furthermore, users are able to browse the explanation facility at any level of depth that seems useful, giving the explanation facility more functionality and the appearance of greater intelligence. Hypertext explanation and browsing facilities might be expanded to allow the expert system to explain itself in a way that is consistent with the explanation that a human expert would give.

4.b Integrating expert system features in a hypertext environment

Hypertext is sometimes difficult to navigate. Expert systems can be embedded in a hypertext environment and serve as an intelligent guide across the links in the text. In one example an "if-then" rule is embedded in a link between textual objects using HyperCard and HyperTalk.

Each card in HyperCard can be viewed as a node in a network. The links connecting nodes can be traversed by clicking the mouse on buttons (see earlier section on embedded conceptual nets). Networks of cards are called stacks. The stack can be browsed by bringing individual cards on the screen and clicking on a button. HyperTalk is the scripting language used to write procedures for HyperCard. One way to integrate expert system capabilities in a HyperCard environment is through HyperTalk.

For example, imagine a card which contains an outline which includes "procedures" as a subtopic under "link types". Each topic in the outline corresponds to a button. If the reader points to the "procedures" button, he is asked whether he has a theoretical computer science training. Depending on the answer to this question, the reader is taken either to an explanation of procedures in terms most meaningful to a theoretical computer scientist or to a general purpose explanation. To achieve this intelligent guiding within HyperCard, the HyperTalk script of Figure 5 could be attached to the "procedures" button.

Consider another example of expertext. One proposed approach to tracking harbour activity involves the daily collection of records on ships crossing the entrance to a bay and records on each pier in the bay (Schutzer, 1985). Each record includes text and a photograph. The harbour computer system might additionally store information about ships in predicates, such as "docked(Titanic, pier 13)". Rules may also be represented in the system to allow the answering of queries by deductive inference. For example, a rule might say "if (arrived(ship y) and empty (pier z) and type-pier(ship y, pier z), then pier-assignment (ship y, pier z)", which basically says that a newly arrived ship will be assigned to the appropriate, vacant pier. If a user of the system is reading about the Titanic's entering the bay yesterday and wants to read about the pier where the ship is likely to be today, then the above rule would support such browsing.

Consider an abstract form for a rule in a hypertext link: "if x then y". The types of rules are a function of the character of x or y. As in the example from HyperCard, x could be a single question which the reader is asked specifically in order to guide a particular step. Thus when the reader is asked whether or not he has a theoretical background (see Figure 5), he will be led to a particular card as a result of his answer. If x is extended to include a sequence of questions, something closer to a traditional expert system is produced. On the other hand, x may depend on previous actions taken by the reader which were not taken for the sake of answering x. For instance, if the reader had chosen at an earlier stage to move to a joke card rather than a poem card, then some later option might be constrained to showing an analysis of a joke rather than an analysis of a poem. For example, in the rule "if reader chose joke earlier, then now go to analysis of joke" x depends on an

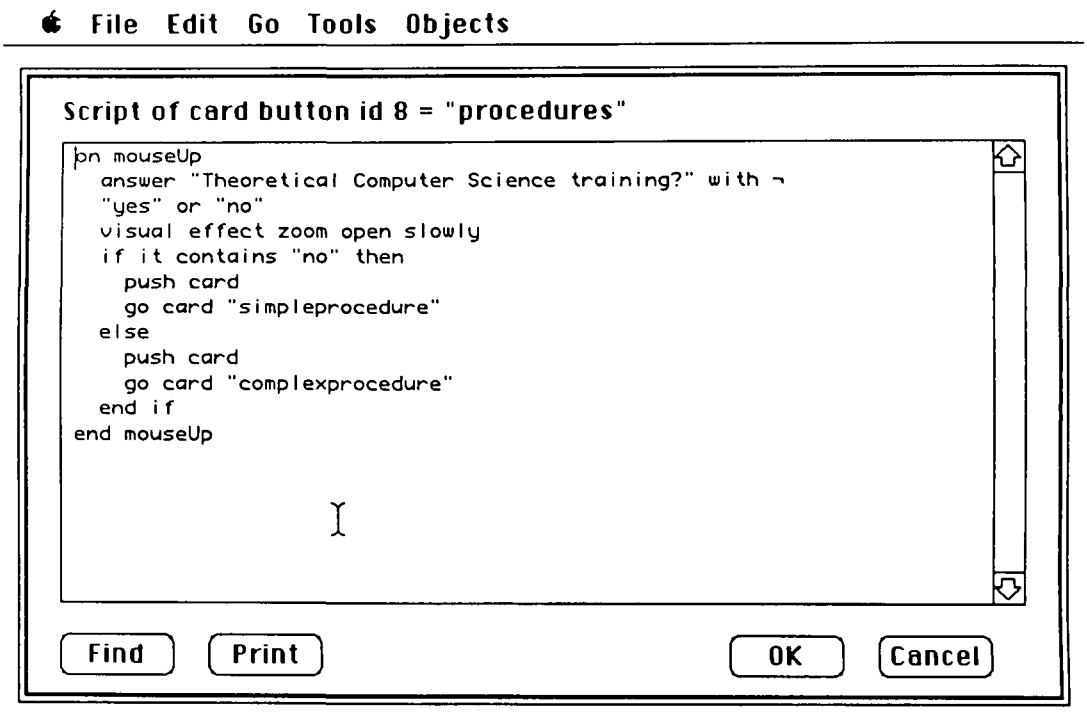

Figure 5 A card which shows the script which was attached to the "procedures" button. This script asks a question and based on the answer decides which card to next bring to the screen

earlier decision of the reader. This type of x can be extended by covering a series of previous decisions of the reader. For instance, if the reader has gone from a certain card C to the help button, back to card C, and again to the help button, then the system might intervene and ask the reader whether he needs to know a way to exit card C other than to go to help.

Varying the nature of the y in "if x then y" creates further classes of hypertext rules. In the discussion of variants of x, y was a card. But y can be more—y could be computed on the fly. For instance, in an electronic yellow pages, the reader may say that he wants to know about Chinese restaurants near his house. The system might then find a small set of restaurant descriptions matching that query and compose from that set a presentation to the reader.

4.c Application to software engineering

Expertext can be applied to *software engineering*. Software engineering involves the creation of a series of documents (Sommerville, 1985). In a certain sense, all of software engineering can be considered in terms of producing and using documents (Garg and Watt, 1987). First the end users specify requirements for a system. Then software engineers translate these requirements into the design of a system. Computer programmers then study the design and create software. Next, test, maintenance, and user manuals are prepared.

A computer program is itself a document, but a special one because both people and machines must understand it. Since programs are inevitably written on computers, computerized writing and reading aids have a ready market. The code and its documentation have many links. Documentation next to a line of code may explain what that line means. For instance, the documentation may say that a certain condition tests whether the salary is within acceptable limits. At a higher level the role of a subroutine is explained, and at another level the purpose of the whole program is summarized. The reader of a program should be able to request to see each of these levels of documentation as the need arises but should not have to see them when instead a survey of the code alone is wanted. A program also has rich connectivity within itself. A variable appears in a do loop, but the definition of the variable is far from the do loop. Yet, a reader of the program may want to see the definition in one window, while reading the do loop in another window.

The multiple author condition of large computer programs means that record keeping as to who

```
procedure cash_register is
      price, vat, total : float;
begin  -- cash_register
      get(price);
       .
       .
      put("Total sale is £");
       .
       .
      get(tendered);
**** undefined identifier
       .
       .
**** unexpected end of file
```

Figure 6 A segment of an ADA program which shows two errors, each flagged with four asterisks

wrote what code and when they wrote it could be useful. One version of code may be appropriate for running with one kind of machine and operating system, while another version of code may be necessary for a different machine and operating system. This historical information could be maintained on a hypertext system.

Honeywell's Los Angeles Development Center (LADC) for documenting the Control Program-Six operating system is a kind of collaborative, hypertext software engineering environment (James, 1985). At LADC, from the moment that a software product is conceived, the responsible programmers record thoughts in a file. As the product solidifies, writers transfer information from programmers and customers into documentation. An assessment of various technical publication departments of computer and industrial firms shows that the average cost to produce camera-ready masters for technical publications is about 250 dollars per page, while at LADC the average cost for the same product is about 80 dollars per page.

Many ongoing projects are applying hypertext strategies to computer programs. The Hypertext Abstract Machine (described earlier) is intended for software engineering applications (Campbell and Goodman, 1988). DIF manages the software life cycle as collaborative hyptertext and is being extended to provide expert assistance to programmers (Garg and Scacchi, 1987).

The kind of expertise which might be associated with a software environment can be illustrated with run-time errors. When the compiler detects an error in syntax, rather than just reporting that an error has occurred, it brings two windows to the screen. One window shows the offending code, and the other window reveals the part of the programmer's manual which describes the correct syntax. In Figure 6, two errors are detected by the ADA compiler. The "undefined identifier" error may both point to the portion of the manual which explains the definition of identifiers and to the portion of the program in which the identifier should be defined. The "unexpected end of file" message may direct the reader to the portion of the manual which explains how to correctly terminate an ADA program and to the beginning of the procedure which is missing the proper ending. Later, after the syntactically correct program has been produced and while it is being tested, intelligent detection of failures could be linked to displays of relevant portions of the documentation.

5 Discussion

The trend towards the uniting of expert systems and hypertext is illustrated by the availability of new software products. For instance, "Knowledge Pro" is a software product that was released in 1988 and which combines expert system and hypertext features. (Furthermore, Knowledge Pro* can be freely copied). One of the files in Knowledge Pro says:

* Information on Knowledge Pro can be obtained from Knowledge Garden, 473A Malden Bridge Road, Nassau, New York 12123

Expert systems don't provide the USER with enough control.
Hypertext systems don't provide the AUTHOR with enough control.
The *knowledge processor* represents the marriage of both techniques
and lets each "react" to the other.

The developers of Knowledge Pro say they have created a "knowledge processor" and that a knowledge processor contains an expert system language and an extensible hypertext system. This combination of expert system and hypertext is what we have called expertext.

In earlier days, clerics performing routine data transactions were the principal users of computer terminals. Now, mid-level, information specialists are finding the computer terminal more attractive, and they are creating and using text and knowledge bases. Making these systems more friendly intelligent is a major challenge to both expert system and hypertext methodologies.

Expert systems represent knowledge so that the computer can act on that knowledge to solve problems. Hypertext systems represent text with links so that the computer can help people create and access a rich textual network. The major shortcomings of expert systems relate to the knowledge acquisition bottleneck and the difficulty of explanation. The major difficulty with hypertext is that people are ill-prepared to explicitly create or access the links between pieces of text.

The knowledge acquisition bottleneck hinders expert system builders. Text is directly meaningful to people. Expertise, to the extent that it is "encoded", is more likely to be represented in documents than in computer rules. Making knowledge available to people through hypertext seems one way to circumvent part of the problem of knowledge acquisition for a system.

Hypertext is sometimes difficult for people to understand. Procedural knowledge in the links could guide people in their document traversal. This kind of expert guidance by the hypertext system means that the author of the hypertext must be explicit about the decisions which the computer will make in response to the decisions by the reader.

Expert system and hypertext methodologies can both be extended so as to give one the functionality of the other. In one approach to building *expertext*, the expert system parts are used for tasks that require deductive inferencing. The expert system modules may access information from the hypertext, but where feasible, routine browsing is handled by the hypertext system. The strengths and weaknesses of expert and hypertext systems so naturally complement one another that expertext—the combination of the two—deserves attention.

In summary, the trends in expert system work are (1) the processing of knowledge from text into expert system and (2) the provision of explanation as text. The trends in hypertext are (1) the adding of knowledge onto text in the form of new links and (2) the embedding of procedures in links to help readers follow the links. These four directions intimately relate one to another and presage the development of knowledge engineering factories to process text into hypertext and then expertext.

6 References

Expert systems

Alty, J L, 1987. "The limitations of rule-based expert systems" In: *Knowledge-Based Expert Systems in Industry*, Jiri Kriz (Ed.), pp. 17–22. A critical review of expert systems which emphasizes their strengths and weaknesses.

Belkin, N, Borgman, C, Brooks, H, Bylander, T, Croft, W, Daniels, P, Deerwester, S, Fox, E, Ingwersen, P, Rada, R, Jones, K, Thompson, R and Walker, D, 1987. "Distributed expert-based information systems: an interdisciplinary approach," *Information Processing and Management* **23** (5) 395–409. A workshop report on the role of expert systems in document retrieval systems.

Benchimol, G, Levine, P and Pomerol, J C, 1987. *Developing Expert Systems for Business*, North Oxford Academic Publishers Limited. A modern introductory text which includes a description of expert system tools and business applications.

Buchanan, B and Shortliffe, E, 1984. "Major lessons from this work," In: *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, ed. E Shortliffe (Ed.), pp. 669–702,

Addison-Wesley, Reading, Massachusetts. This chapter from the book about the MYCIN experiments summarizes the major insights that were gained. The authors are the fathers of the MYCIN experiments.

Carroll, John and Aaronson, Amy, 1988. "Learning by doing with simulated intelligent help," *Communications of the Association of Computing Machinery* **31** (9) 1064–1079, September. The difficulties in designing intelligent help and explanation facilities are addressed.

Clancey, William, 1983. "The epistemology of a rule-based expert system—a framework for explanation," *Artificial Intelligence* **20** (3) 215–251. An early influential paper on the need for additional knowledge in MYCIN in order that explanation be satisfying.

Forsyth, Richard and Rada, Roy, 1986. *Machine Learning: Expert Systems and Information Retrieval*, Ellis Horwood, London. Methods for semi-automatically improving the performance of intelligent systems are described.

Gallaire, H, 1985. "Logic programming: further developments," In: *Proceedings 1985 Symposium on Logic Programming*, pp. 88–96. Meeting occurred in July 1985 in Boston. A historical overview of logic programming and a critical analysis of its present and future applications.

Hayes-Roth, F, Waterman, D and Lenat, D, 1983. *Building Expert Systems*, Addison-Wesley, Reading, Massachusetts. This book contains chapters written by participants in a workshop on building expert systems. The target audience includes practitioners and administrators.

Mili, Hafedh and Rada, Roy, 1988. "Merging thesauri: principles and evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **10**(2) 204–220. The system from the National Library of Medicine is discussed with respect to knowledge acquisition.

Miller, R, Pople, H and Myers, J, 1982. "INTERNIST-1," *New England Journal of Medicine* **307**, pp. 468–476. A report to physicians on the successes with the expert system INTERNIST.

Rada, Roy, 1985. "Gradualness facilitates knowledge refinement," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **7**(5) 523–530, September. Methods for allowing small changes in the weights on rules of an expert system to lead to improved performance of the expert system are presented.

Schutzer, Daniel, 1985. "Artificial intelligence-based very large data base organization and management," In: *Applications in Artificial Intelligence*, Stephen Andriole (Ed.), pp. 251–277, Petrocelli Books. An application for large intelligent heterogeneous databases is clearly described.

Shortliffe, E. Scott, A, Bischoff, M, Campbel, A, van Melle, W and Jacobs, C, 1981. "ONCOCIN: expert system for oncology protocol management," *Proceedings International Joint Conference Artificial Intelligence*, pp. 876–881. One of the first published papers on the ONCOCIN project which represents the intellectual and pragmatic descendant of MYCIN.

Slatter, Philip E, 1987. "Cognitive emulation in ES design," *The Knowledge Engineering Review* **2**(1) 27–42, March. A study of how cognitive emulation can be used as a strategy for designing more effective expert systems.

Stenton, S P, 1987. "Dialog management for co-operative knowledge based systems," *The Knowledge Engineering Review* **2**(2) 99–122. A review of user interaction with knowledge-based systems and approaches for improving communications between users and expert systems.

Timpka, Toomas, 1986. "LIMEDS knowledge-based decision support for general practitioners: an integrated design," *Proceedings Tenth Annual Symposium on Computer Applications in Medical Care*, pp. 394–402, IEEE Computer Society, October. The Scandinavians have implemented some sophisticated, intelligent, hypertext systems in their ambulatory care facilities.

Waterman, Donald A, 1985. *A Guide to Expert Systems*, Addison Wesley. An introductory text which includes an overview of expert system tools and applications.

Zaniolo, Carlo, Ait-Kaci, Hassan, Beech, David, Cammarata, Stephanie and Maier, David, 1986. "Object oriented database systems and knowledge systems," In: *Expert Database Systems*, Larry Kerschberg (Ed.), pp. 49–65, Benjamin/Cummings Publishing. This report from a workshop covers many of the major relationships between object-oriented databases and knowledge bases.

Hypertext

Acksyn, R, McCracken, D and Yoder, E, 1988. "KMS: a distributed hypermedia system for managing knowledge in organizations," *Communications of the Association of Computing Machinery* **31**, 7, pp. 820–835.

Brown, P J, 1987. "Turning ideas into products: the guide system," *Hypertext '87*, pp. 33–40, University of North Carolina, Chapel Hill, North Carolina, November 13–15. The Guide system focuses on a few simple link types, and this paper by the developer of Guide cogently argues why that approach is a good one.

Bush, Vannevar, 1945. "As we may think," *The Atlantic Monthly*, pp. 101–108, July. The first widely read paper to seriously predict the advent of hypertext.

Campbell, Brad and Goodman, Joseph M, 1988. "HAM: a general-purpose hypertext abstract machine,"

Communications of the Association of Computing Machinery **31**(7) 856–861 (also appeared in Hypertext '87). The HAM system allows ready simulation of many hypertext systems, as this paper shows.

Collier, George, 1987. "Thoth-II: hypertext with explicit semantics," *Hypertext '87*, pp. 269–289, University of North Carolina, Chapel Hill, North Carolina, November 13–15. The independent and embedded link networks of hypertext are described in this analysis of different types of networks in hypertext.

Conklin, Jeff, 1987. "Hypertext: an introduction and survey," *Computer* **20**(9) 17–41, September. This introduction to hypertext was one of the first comprehensive, scholarly review of this discipline to appear in a major computer journal.

Englebart, D C and English, W K, 1968. "A research center for augmenting human intellect," *Proceedings of the Fall Joint Computer Conference, 33*, pp. 395–410, AFPIS Press, Reston, Virginia. The senior author pioneered the implementation of hypertext systems.

Garg, Pankaj K 1988. "Abstraction mechanisms in hypertext," *Communications of the Association of Computing Machinery* **31**(7) 862–870. This model of hypertext has more definitions than theorems but is noteworthy for attempting to model hypertext and for appearing in a widely available journal.

Goodman, Danny, 1987. "Hypercard," *MacIntosh Today*, pp. 60–64, August 11. Hypercard was first made publicly available in the summer of 1987 and this paper was one of the early, informative descriptions of the system.

Halasz, Frank G, Moran, T P and Trigg, Randal H, 1987. "NoteCards in a nutshell," *Proceedings of the ACM CHI+GI Conference*, pp. 45–52, Toronto, Ontario, April. A description of the NoteCards system.

Halasz, Frank G, 1988. "Reflections on NoteCards: seven issues for the next generation of hypermedia systems," *Communications of the Association of Computing Machinery* **31**(7) pp. 836–855. Beginning with a critique of the first, major commercial hypertext system, the author then carefully delineates seven major problems which researchers in hypertext must address over the next generation.

Hayes, John R, Flower, Linda, Schriver, Karen A, Stratman, James F and Carey, Linda, 1987. "Cognitive processes in revision," In: *Advances in Applied Psycholinguistics*, Sheldon Rosenberg (Ed.), pp. 176–240, Cambridge University Press, Cambridge, England. The authors have performed some of the best recognized work in the psychology of writing and here describe their model of writing with an emphasis on the revision process.

Koved, Larry and Shneiderman, Ben, 1986. "Embedded menus: selecting items in context," *Communications of the Association for Computing Machinery* **29**(4) 312–318. Careful human-factors analyses are used to support arguments for a certain interface style with a hypertext system.

McCracken, Donald and Akscyn, Robert, 1984. "Experience with the ZOG human-computer interface system," *International Journal of Man-Machine Studies* **21** pp. 293–310. A decade of experience with ZOG is summarized.

Mili, Hafedh and Rada, Roy, 1988. "Merging thesauri: principles and evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **10**(2) 204–220. The system from the National Library of Medicine is discussed with respect to knowledge acquisition.

Price, Lynne A, 1982. "Thumb: an interactive tool for accessing and maintaining text," *IEEE Transactions on Systems, Man, and Cybernetics* **12**(2) 155–161, March/April. An early project for loading traditional documents into a hypertext system and for adding links to the document so that browsing can follow more paths.

Raskin, Jef, 1987. "The hype in hypertext," *Hypertext '87*, pp. 325–330, University of North Carolina, Chapel Hill, North Carolina, November 13–15. The difficulties of finding practical applications for hypertext are outlined in this paper.

Shneiderman, Ben and Morariu, Janis, 1986. "Design and research on the interactive encyclopedia system (TIES)," *Proceedings 29th Conference of the Association for the Development of Computer Based Instructional Systems*, pp. 19–21, November. TIES is another one of the academic hypertext systems that has been modestly successful in the commercial world.

Smith, John B, Weiss, Stephen F, Ferguson, Gordon F, Bolter, Jay D, Lansman, Marcy and Beard, David V, 1986. "WE: A Writing Environment for Professionals," *Technical Report TR86-025*, Department of Computer Science, University of North Carolina, Chapel Hill, North Carolina, August. Innovative early work on the testing of a writing environment on a graphical workstation.

Stotts, P David and Furuta, Richard, 1988. "Adding browsing semantics to the hypertext model," *Technical Report CS-TR-2046*, Computer Science Department, University of Maryland, College Park, Maryland, June. As hypertext has become more popular, theoretical models are beginning to appear, and this report is one of the best early models.

Walker, William and Kintsch, Walter, 1985. "Automatic and strategic aspects of knowledge retrieval," *Cognitive Science* **9** 261–283. Memory retrieval experiments show that the representational schemes focus on episodes.

Weyer, Stephen A and Borning, Alan H, 1985. "A prototype electronic encyclopedia," *ACM Transactions on Office Information Systems* **3**(1) 63–88. Description of a system for translating an encylopedia into hypertext and browsing the hypertext. While the particular project has stopped, the ideas are sound.

Zaniolo, Carlo, Akscyn, Robert, McCracken, Donald and Yoder, Elise, 1988. "KMS: a distributed hypermedia system for managing knowledge in organizations," *Communications of the Association of Computing Machinery* **31**(7) 820–835 (also appeared in Hypertext '87). KMS is the descendant of ZOG which was developed in the 1970s and has been extensively used at Carnegie-Mellon University.

Software engineering

Garg, Pankaj K and Scacchi, Walt, 1987. "On designing intelligent hypertext systems for information management in software engineering," *Hypertext '87*, pp. 409–432, University of North Carolina, Chapel Hill, North Carolina, November 13–15. Issues underlying the implementation of an intelligent hypertext software engineering tool.

James, Geoffrey, 1985. *Document Databases*, Van Nostrand Reinhold Company, New York. An unpretentious book with many examples that focuses on a software hypertext system that is actually used.

Sommerville, Ian, 1985. *Software Engineering*, Addison-Wesley. A standard textbook on software engineering.