Structured Hypertext with Domain Semantics

WEIGANG WANG
GMD-IPSI: German Research Center for Information Technology
and
ROY RADA
Pace University

One important facet of current hypertext research involves using knowledge-based techniques to develop and maintain document structures. A semantic net is one such technique. However, most semantic-net-based hypertext systems leave the linking consistency of the net to individual users. Users without guidance may accidentally introduce structural and relational inconsistencies in the semantic nets. The relational inconsistency hinders the creation of domain information models. The structural inconsistency leads to unstable documents, especially when a document is composed by computation with traversal algorithms. This work tackles the above problems by integrating logical structure and domain semantics into a semantic net. A semantic-net-based structured-hypertext model has been formalized. The model preserves structural and relational consistency after changes to the semantic net. The hypertext system (RICH) based on this model has been implemented and tested. The RICH system can define and enforce a set of rules to maintain the integrity of the semantic net and provide particular support for creating multihierarchies with the reuse of existing contents and structures. Users have found such flexible but enforceable semantics to be helpful.

Categories and Subject Descriptors: E.1 [**Data**]: Data Structure—*graphs*; H.2.1 [**Database Management**]: Logical design—*data models*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software; H.5 [**Information Systems**]: Information Interfaces and Presentation; 1.7.m [**Text Processing**]: Miscellaneous—*hypertext*

General Terms: Design, Documentation, Management

Additional Key Words and Phrases: Graph theory, hypertext models, hypertext structures

1. INTRODUCTION

The essence of hypertext is the computer-supported link [Conklin 1987]. The links provide the structure and access means of a hypertext. The

Authors' addresses: W. Wang, GMD-IPSE, Doliovostrasse 15, Darmstadt 64293; R. Rada, School of Computer Science and Information Systems, Pace University, Pleasantville, NY 10570.

structure formed by links and nodes can be seen abstractly as a graph, whose subgraphs can be of many types, such as a network, a tree, or a directed acyclic graph (DAG). The logical structures in hypertext are usually supported by means of composites (a group of nodes and links) [Halasz and Schwartz 1994]. The meaning of a link is often indicated by the semantic type of the link. Link types can help users and computers to distinguish various kinds of links, i.e., whether a link is a traversal connection, a structural means, or an argument representation [Trigg 1996].

With the widespread use of the WWW comes renewed interest in the semantics of hypertext. Bill Gates in his Comdex'96 keynote lecture said the following about the WWW:

> We also need to take links and give them types, so that if I go to a site and say I want to use this site offline, it can understand based on the link types what pages to bring down. And so we need new standards in that area. The standards committees are working very well here—W3C, IETF—and so I think this richer structure will come quite rapidly [Gates 1996].

Numerous systems with typed links were first developed before the WWW existed, and then the ideas were ported to the WWW. Note for instance the structured editing facilities in the work of Decouchant et al. [1996] or the Petri net semantics of Stotts et al. [1996]. Our collaborative hypertext system was developed over 10 years and shows the role of linking semantics.

Hypertext may be profitably viewed as a semantic net. A semantic net is a directed graph in which concepts are represented as nodes and relations between concepts are represented as links. The analogy of a semantic net to hypertext has long been recognized, and a semantic net has been considered as a logical model of hypertext, especially for those hypertexts with typed nodes and links [Conklin 1987].

Hypertext systems can be classified based on the structural and relational constraints on the semantic net. The structural constraints concern the *shape* of a subnet, such as a tree or a DAG. The relational constraints concern the *meaning* of semantic typing, for example, to express the meaning of "an issue generates a position"; an "issue"-type node can be connected with a "generate"-type link to a "position"-type node in an issue-based system [Conklin and Begeman 1989]. We consider three types of semantic-net-based hypertext systems:

—An *unstructured hypertext* system can have arbitrary node and link types the use of which is free from constraints. The preservation of any patterns depends wholly on voluntary actions of the authors.
—A *semistructured* hypertext system provides a set of recommended link types, but relies on users' voluntary actions to implement the constraints.
—A *structured hypertext* system has rules for structural and relational constraints, and these constraints are enforced by the hypertext system.

In Sections 1.1 and 1.2, first, the problems with *unstructured* and *semistructured* hypertext are analyzed through two typical examples of such systems. Then, the *structured hypertext* approach to the problems is outlined.

1.1 Problem Analysis

The widespread use of the Web makes hypertext a common concept for ordinary users. However, the hypertext provided by the current Web is largely an *unstructured hypertext.* In the current Web, links are not typed, and there is no composition mechanism. Thus the current Web lacks structure [Trigg 1996]. When a large document is contained in a single Web page, the hypertext look and feel and its benefits are reduced. When a document is split into multiple Web pages (files), it is hard for users and computer systems to tell whether the target of a link is a logical part of the current document or referential material related to the document. Often users want to get a hard copy of an entire document they find on the Web, but unless the document is contained in a single Web page, they have to print it manually, page by page. This is not convenient. Some programs on the Web fetch documents. The programs begin at some Web file and then recursively follow every link. These programs typically find many files that are not semantically an integral part of the first file.

The authors were devoted to the development of the MUCH (Many Using and Creating Hypertext) system in the late 80's and the early 1990's [Rada et al. 1991; 1992; Wang and Rada 1995]. The experience from several prototypes of the MUCH system led to the work presented in this article. The MUCH system is based on the Dexter model [Halasz and Schwartz 1994] and treats the storage layer as a semantic net [Lehmann 1992]. The MUCH system provides a number of recommended link types for representing application domain concepts, such as thesauri, documents, and annotations. Users of the system were expected to use those link types in the course of authoring meaningful hypermedia. The various functionalities of the system then exploit the knowledge in this semantic net. The standard perspective is of a fold-unfold outline (a hierarchical view) of the semantic net. Traversals of the net with various filters are the basis for the user's views of the semantic net. While the logical model of the MUCH system is a semantic net, a user sees at any time a view of this net as a tree. The tree (an indented list of node names, or an outline) can be folded and unfolded. When a user selects a node in the fold-unfold outline, the associated content appears in the content window. The hypertext provided by the MUCH system is a kind of *semiunstructured hypertext*. In the MUCH system, linking is not constrained by the system. Links of any type can be used to form a tree or a network.

The MUCH system was used between 1991 and 1994 by over 200 people. Users of the system are expected to use link types in the course of authoring meaningful hypermedia. However, contrary to the expectations of the builders of the MUCH system, the users did not exploit the ability to

type semantic links. Typically authors used the default link type regardless of their semantic intentions. When a link type other than the default type was chosen, that choice was often inconsistent with the way another user would label a similar link. The system has proven to be useful for authoring conventional documents. Authors, however, were not practically able to produce hypertext documents [Wang and Rada 1995].

One could argue that if the users had understood the "simple" depth-first traversal that was used to generate the hierarchical view (outline), then the problems with document composition would disappear. Experience, however, suggested that users did not appreciate the effects of the traversal algorithm, which requires them to understand the consequences (changes of hierarchical views) caused by adding, deleting, or "deadending" a link between two existing nodes. For those who may be able to predict the outline traversed from a small network, they might fail to do so when the numbers of nodes increases and when multiple users are modifying the network at the same time. Users wanted "stable" documents. Other views upon the same document can be appreciated only after the stable view exists. Documents in hypertext should be both fixed (stable) and fluid (changeable), rather than fluid only [Levy 1994]. The problems with document composition (unstable view) and linking inconsistency may be rooted in the hypertext data model of the MUCH system. The model is a labeled directed graph (a semantic net), but has no structural or relational constraints on the construction of the net. Although the MUCH system provides four link types, it does not check whether users follow any consistency constraints on the use of these link types. KMS also uses a depth-first traversal algorithm to generate hierarchical views for printing purposes [Akscyn et al. 1988]. KMS has no such problem, because it traverses only tree links whose structure is maintained by the system. Results from previous work suggests that hypertexts built without system-enforced constraints will fail to manifest predictable patterns of structure or semantics [Wang and Rada 1995].

1.2 Our Approach

This work tackles the above problems by integrating logical structure and domain semantics into semantic nets. We propose a semantic-net-based *structured-hypertext* model that captures common structural and semantic characteristics of technical documents. The method for creating such a structured hypertext has the following four components:

—domain models that depict the domain concepts and relations among those concepts,

—semantic net representation of the domain model,

—rules to govern the description of the semantic net, and

—operators for manipulating the semantic net.

This structured hypertext with domain semantics is a semantic data model for our targeted application domain.

The remainder of this article is organized as follows. Section 2 gives the domain model and requirements for the structured hypertext. Section 3 develops the domain model and requirements into a formal semantic data model. Section 4 presents the Reusable Intelligent Collaborative Hypertext (RICH) system which is developed using the model (this section reveals a flavor of what the system is supposed to do from the user's point of view). Section 5 provides some initial user studies of the RICH system. Section 6 compares this work with related work, and Section 7 concludes the article.

2. DOMAIN ANALYSIS

Domain analysis is a method for analyzing an application domain by studying and characterizing existing systems, underlying theory, domain expertise, emerging technology, and development histories within a domain of interest [Lung and Urban 1993]. In domain analysis common characteristics from similar systems are generalized; objects and operations common to all systems within the same domain are identified; and a domain model is defined to describe the relationships between the objects [Prieto-Diaz and Freeman 1987]. The application domain for this work is technical document authoring with intensive reuse of existing documents. The systems exploited are semantic-net-based hypertext systems. The results of the analysis are a domain information organizational model and requirements for structured hypertext.

2.1 Domain Models

For effective use and reuse, information needs to be well organized. In the technical document application domain, hierarchical structures are widely accepted logical structures for documents. The hierarchical structure as demonstrated in the classification scheme of a library and in the outline of a single document is a well-established model. In a thesaurus, the "narrower than" (NT) relation forms a hierarchical classification structure (among "Concepts"), and the "related" (RT) and "used for" (UF) relations are attached to the hierarchical structure. In a document, a table of contents (or an outline) helps readers and writers to visualize the structure of the document. The relation between section titles and subsection titles is a kind of NT relation, and the cross-reference relation is a kind of RT relation.

The comparison between thesauri and documents highlights the similarity between a library and a document: they all have a skeleton consisting of the NT and RT kinds of relationship. On the basis of this similarity, a hypertext organization model (a structured-hypertext model) is derived. In this model, the hierarchical (or multihierarchical) relation of the thesaurus and the table of contents constitutes a backbone, while referential relations are attached to the backbone to form a network. Links in the model are classified into two categories: organizational links and referential links. The organizational links are used for hierarchical or multihierarchical relations. They are constrained to form a rooted ordered DAG. The referential links have no structural constraints. The clear differentiation of organizational relationship with referential relationship can be found in

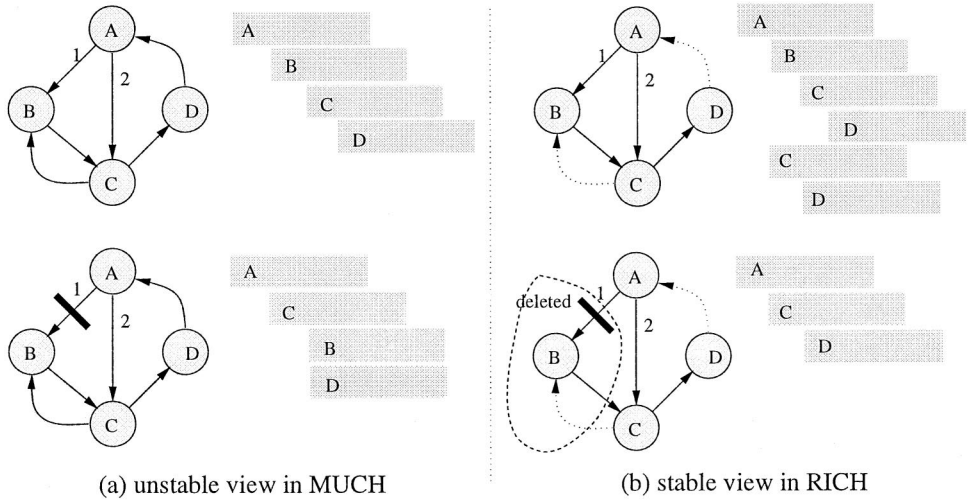

Fig. 1. Hierarchical views of a graph.

SGML-based structured documents and in many hypermedia systems. The typical examples are tree links and cross-reference links in KMS and structural links and hyperstructural links in DGS [Shackelford et al. 1993].

There are also differences in the structures of thesauri and structured documents. Documents usually have much richer relationships and need a a much higher level of cohesiveness than what thesauri have. Therefore, links can be further classified into more detailed semantic types. The use of the typed links should be constrained to conform with the domain model, e.g., the thesaurus NT link is supposed to be used between two Concept nodes, and the thesaurus UF link is supposed to be used between a thesaurus Concept and a thesaurus Synonym.

By classifying links into organizational and referential categories, and by incorporating the structural constraints (i.e., the DAG) on organizational links, the problems with document composition (i.e., the unstable hierarchical views of the MUCH system) can be solved. This can be illustrated by Figure 1, which compares the document composition methods that are used in the MUCH system and its successor, the RICH system. In Figures 1(a) and (b), the graph on the upper left-hand side is traversed (starting from node A) to produce the hierarchical views on the upper right-hand side. In the graph on the lower left-hand side, the thick bar indicates a deletion of link. This deletion leads to the hierarchical view (a list of indented headings) shown on the lower right-hand side. As shown in Figure 1(a), to generate a hierarchical view, the MUCH system traverses links of any type (using a depth-first traversal algorithm [Carre 1979; Wang and Rada 1995]). The deletion of a link results in a hierarchical view drastically different from the initial view (i.e., the hierarchical relations among node A, B, C, D are changed). As shown in Figure 1(b), to generate a hierarchical view, the RICH system traverses (by default) only organizational links (indicated by the straight arrows); the referential links (indicated by curve

dotted arrows) are ignored. The deletion of an organizational link also causes the deletion of some other links and nodes that become inaccessible along organizational links from other existing nodes (as seen in the dotted circle), and in the resultant hierarchical view, the hierarchical relations among existing nodes (in this case, nodes A, C, D) remain unchanged.

The constructing method for generating stable hierarchical views in the RICH system is based on a variant of depth-first traversal. The traversal starts from a given node, and then goes along organizational links (by default) to produce a hierarchical backbone. In addition to differentiating links, the traversal algorithm of the RICH system differs from the depth-first traversal that is used in the MUCH system in that a node is allowed to be visited more than once. The link categorization and the DAG structure of organizational links ensure that endless recursion would not happen, although some nodes or hierarchies may appear more than once. The duplication of appearance is intentionally allowed for presenting multihierarchical views. Comparing Figure 1(a) with Figure 1(b), it can be seen that in the MUCH system the deletion or creation of links may dramatically change the hierarchical views, while in the RICH system the link deletion and creation will bring only small changes that are easy to appreciate.

The problem with linking consistency (the inconsistent use of typed links) can be solved by incorporating semantic linking rules that are implied in domain models. A semantic linking rule can be defined as a triple ⟨source node type, link type, target node type⟩, which specifies an allowable typed link between two typed nodes. For instance, from a node of type A, a type B link to a node of type C is allowed to be created, only if there exists a linking rule ⟨A B C⟩.

The manipulation operators of the structured hypertext should be defined in a way that they would not destroy the semantic integrity of the semantic net. The semantic integrity of the semantic net is reflected in the structural integrity of the semantic net (a semantic net has structural integrity, if, and only if, the structural constraints are maintained) and the relational integrity of the semantic net (a semantic net has relational integrity, if, and only if, the relational constraints are maintained). The semantic knowledge reflected in the structural and the relational constraints defines a domain model for organizing information in a meaningful way. This structure together with its domain-specific semantics constitutes a kind of structured hypertext.

Despite having the above problems that we try to tackle with a structured hypertext, the MUCH system does support a restricted domain model. To build and maintain semantic nets for hypertext, indexing and semantic net construction go hand-in-hand. Indexing occurs when a document (or a document component) is attached to the semantic net. If a document (or a document component) is to be added to the document collection but no component of the semantic net relates to it, then the semantic net is augmented. The recommended method for constructing a thesaurus is to start with a document collection, to index documents, and to create terms for the thesaurus as needed for indexing [Soergel 1974]. In

this sense, the construction method for the semantic net of a document is similar to the construction method for thesauri. The text associated with document nodes must give rise to a cohesive story in a way that text associated with thesaurus nodes does not. The similarities and differences between the semantic net components for documents and thesauri must be appreciated in the design of a well-founded hypertext [Mili and Rada 1988].

As the construction of thesauri and semantic nets can be a very time-consuming exercise, tools to facilitate such construction have been developed. The word-frequency method is based on the widely accepted hypothesis that words (or terms) that often cooccur in textual documents tend to be linked according to some relationship important to the document space [Lesk 1969]. The MUCH system includes various programs that automatically select index terms from text blocks, make nodes of these terms, and then create links between these nodes or terms [Rada 1990; 1992].

2.2 Document Reuse Processes

Document reuse occurs when information created for one purpose (in one document) is used for another purpose (in a new document). The modularity of information units and the flexibility in organizing these units make hypertext appropriate for document reuse. Document reuse in hypertext can be seen as a practice in which contents and structures created for one document (or one view of a document) are used for a new document (or another view of a document). Thus, the content of a node can be reused by another node. New structures can be generated from the existing structures of the underlying hypertext. The central idea of Xanadu, pioneered by Ted Nelson, has been reuse through embedded shared instances [Nelson 1995]. Overlapping structures supporting multiple views is a feature of hypertext at its best [Trigg 1996]. Therefore, many hypertext structures, such as paths [Zellweger 1989], guided tours [Marshall and Irish 1989], composite nodes [Halasz and Schwartz 1994], and views [Tomek et al. 1991], can be used to support document reuse. In the hypertext literature the approaches to building these structures tend to be quite general, for example approaches based purely on graph theory. The general approaches, however, are not helpful in supporting document reuse. The systems based on general approaches tend to have no or only very limited understanding of the data they process [Evenson et al. 1989]. They do not support the semantic structuring of the hypertext as the hypertext is authored. If the consideration for document reuse can be made when the hypertext is first created, then hypertext can create an environment where reuse is an integral part of the authoring process.

We take a process-based approach to document reuse. The document reuse processes can be decomposed into a series of subprocesses, such as creating, managing, retrieving, and reorganizing processes. The document creation processes capture, organize, and represent knowledge about a domain, and use that knowledge to develop reusable document components that can be used to produce new documents. Document management

processes acquire, evaluate, and organize documents. Central to document management is a document library. Document retrieval processes browse and search the library and the working space to find reusable components. Document reorganization processes tailor the found document components and integrate them into a new document. Tailoring is a modification process. Reusable components typically require at least some modification during reuse. Integration processes ensure that reusable components that have been selected and tailored are properly integrated with other components of the document. Good information representation and organization can make the retrieval process easier, and good representation and presentation can make the understanding and reorganization (modification and integration) process easier.

The document reorganization process may involve

—*copying without modification:* an existing document component is used in a new document without modification, or

—*copying with modification:* an existing document component is used in a new document with changes, or

—*referring:* an existing document component is referenced in a new document, but not included as a logical part of the new document, or

—*sharing:* an existing document component is included as a shared logical part of both the existing document and a new document.

The effect of copying is to duplicate the appearance of an item. Normally, the duplication of appearance results from the real duplication of the copied item. While in hypertext, in addition to a hard copy, duplicated appearances can also be achieved through sharing and referring. Modification is performed by means of adding, deleting, or moving. Reuse through hard copying and modification is common. Reuse through referring and sharing is a feature of hypertext at its best. In practice, these patterns are often combined to suit particular situations.

The above characterization leads to the definition of some crucial requirements for hypertext-based document reuse. Corresponding to the four categories of document reuse processes, these requirements can be classified into four categories:

(1) On document creation, a rigorous hypertext data model for representing and manipulating documents should be provided, in which
   —content should be separate from structure (to make information modules reusable in more than one context),
   —constructs for hierarchical/multihierarchical structures should be provided (for creating logical structures of documents and for creating classification structures of a document library),
   —cross-referencing should be supported (for creating associative relations within a document, among different documents or classification structures),
   —a set of primitive operators (such as addition, deletion, and modification) for manipulating the data of the data model should be provided,

—the structural constraints defined in the data model should be maintained after any operator is performed, and
—the relational constraints reflected in the information organizing structure should be maintained after any operator is performed.

(2) On document management, an information organizing model should be provided, in which
—a document library has a hierarchical/multihierarchical classification structure.
—after being used or reused, the documents in the library are intact,
—a working document space for unfinished documents is provided, and
—a versioning mechanism is provided.

(3) On document retrieval, an information retrieving mechanism should be provided and
—users should be able to read logically structured parts of a document in conjunction with the reading of related information and
—searching should be provided to help users find the materials they want.

(4) On document reorganization, in order to facilitate the above-mentioned different reorganization patterns, a mixture of sharing and nonsharing of changes on content and structure should be allowed, such that
—sharing changes on content and structure (this allows users to share a document component or refer to a document component),
—sharing changes on content but not on structure (this allows users to create multiple views or perspectives on the same set of contents), sharing changes on structure but not on content (this corresponds to a kind of template, which allows users to propagate structural changes to all instances), and
—not sharing changes on content and structure (this produces a hard copy of a document component).

More justification for these requirements can be found in the evaluative study of Wang [1995]. These requirements provide bases for the data structures, rules, and operators of the formal model presented in the next section.

3. FORMAL SPECIFICATION OF STRUCTURED HYPERTEXT

There is a reasonable body of work dealing with formal mechanisms to capture structure and modification of hypertext. One approach is based on set theory [Parunak 1991]. Another approach is based on graph theory [Afrati and Koutras 1990; Bench-Capon and Dunne 1989; Garg 1988; Koo 1989]. The third approach is based on automaton theory [Stotts and Furuta 1989; Stotts et al. 1992]. In the set-based approach, taxonomical reasoning is the foundation for describing functions of hypertext. In the graph-based approach, nodes and links are used extensively in the description of functions of hypertext. The automaton-based approach emphasizes inherent document structure with browsing semantics. Formal specification languages, such as Z [Spivey 1989] and VDM [Ince 1990], are often used to specify a hypertext model that may be based on above-mentioned ap-

proaches [Chua and Lai 1991; Halasz and Schwartz 1990; Tochtermann and Dittrich 1995]. Formal specification languages are used to describe what a system must do without saying how it is to be done.

In this work the graph-based approach is adopted. In this work hypertext is considered as a "directed graph" or "semantic net," and an emphasis is placed on various structures that are formed with nodes and links. These structures can be readily described with existing graph notations, such as "tree" and "acyclic graph." The graph notations not only can define a model that specifies *what a hypertext system must do*, but also have corresponding graph algorithms that specify *how it is to be done*.

The focus of this section is to formalize the domain model and requirements described in the previous section into a rigorous data model so as to provide a solid basis for the RICH system. In the following subsections, first, a brief introduction to graph notations is given. Then the data model of the structured hypertext is defined. Finally, the formal models for presenting the structured hypertext are given. In conjunction with the formal definitions, natural language explanations are provided.

3.1 Graph Notation

A directed graph (or digraph) G = (N, L) consists of

(1) a finite nonempty set of elements called *nodes* (or vertices) N = $\{n_1, n_2, \ldots, n_n\}$ and
(2) a subset L of the Cartesian product $N \times N$, the elements of which are called *links* (or edges).

A link from node n_i to node n_j is denoted as $\langle n_i, n_j \rangle$. A *path* is a finite sequence of links of the form

$$\mu = \langle n_{i_0}, n_{i_1} \rangle, \langle n_{i_1}, n_{i_2} \rangle, \ldots, \langle n_{i_{r-1}}, n_{i_r} \rangle$$

or is denoted as a sequence of the endpoints of the links:

$$\mu = n_{i_0}, n_{i_1}, n_{i_2}, \ldots, n_{i_{r-1}}, n_{i_r}$$

A path whose endpoints are distinct is said to be open; whereas a path whose endpoints coincide is called a closed path, or a *cycle*.

In a graph G = (N, L), a node n_j is called a *successor* of node n_i if $\langle n_i, n_j \rangle \in L$; the set of all successors of n_i is denoted by $\Gamma^+ (n_i)$. Similarly, a node n_j is called a predecessor of n_i if $\langle n_j, n_i \rangle \in L$, and the set of all predecessors of n_i is denoted by $\Gamma^- (n_i)$. Given $n_i, n_j \in N$, if there is a path from n_i to n_j, then n_j is called a *descendant* of n_i; while if there is a path from n_j to n_i, then n_j is called an *ascendant* of n_i. A node n_j is said to be *accessible* from a node n_i if n_j is a descendant of n_i or $n_j = n_i$; similarly, n_j is said to be *converse-accessible* from n_i if n_j is an ascendant of n_i or $n_j = n_i$. The set of nodes which are accessible and converse-accessible from n_i are denoted by $\Gamma^{*+} (n_i)$ and $\Gamma^{*-} (x_i)$, respectively. The number of links going out from node n_i is called the *out-degree* of n_i and is denoted by ρ^+

(n_i); and the number of links coming into node n_i is called the *in-degree* of n_i and is denoted by $\rho^-(n_i)$.

An acyclic graph is a graph which does not contain any cycles. Any acyclic graph contains at least one node which has no successors, and at least one node which has no predecessors. A necessary and sufficient condition for a graph G = (N, L) to be acyclic is that its nodes $n_1, n_2, \ldots, n_n$ can be numbered (i.e., assigned their indices) in such a way that if $\langle n_i, n_j \rangle \in L$ then $i < j$ [Carre 1979].

$G' = (N', L')$ is called a subgraph of a directed graph $G = (N, L)$, if G' is also a graph, and $N' \subseteq N$ and $L' \subseteq L$. All the notions defined above on graph G, such as path and cycle, also applies to its subgraphs. A subgraph provides a more specified context for these concepts. For instance, a path in a subgraph of G is also a path in G; while a path in G may not be a path in some of its subgraphs.

A semantic net is a type of directed graph, in which nodes and links are labeled with semantic meanings. A link in a semantic net is typically represented as a triple of ⟨source node, link type, target node⟩. For example, between two nodes labeled as Birds and Animals, a link could be ⟨Birds, are, Animals⟩. Therefore, a semantic net can be denoted as G = (N, T, L), where N for nodes, T for link types, and $L \subseteq N \times T \times N$ for links consisting of the triples.

3.2 Semantic Net Representation

A data model consists of two elements [Ullman 1983]:

(1) a mathematical notation for expressing data and relationships and
(2) operations on the data that serve to express queries and other manipulations of the data.

In addition to these two elements, a semantic net type semantic data model has constructs for high-level application domain concepts, and rules that govern the description of the domain concepts [Potter and Trueblood 1988; Parsaye et al. 1989; Schnase et al. 1993].

The semantic data model of the structured hypertext H_s is defined as

$$H_s = (G, M_{nw}, R_s, O)$$

where G is a directed graph representing the structured hypertext. M_{nw} is a composition scheme for constructing composite structures. R_s is a set of rules that govern the description of domain concepts. O is a set of operations on the structured hypertext. The definition of the four components are provided in the following four subsections.

3.2.1 *Graph Structure.* G is a directed graph representing the structured hypertext. It is defined as a six-tuple: $G = (N, L, C, P, T, S)$, in which N is a set of nodes. L is a set of links. C is a set of Contents. Nodes, Links, and Contents are data objects which may have attributes; each

attribute is denoted as *dataobject.attribute*. P matches a node to the node content. T is a set of link types, and S is a set of node types.

$$G = (N, L, C, P, T, S), \text{ where}$$

$$N \neq \emptyset,\ C \neq \emptyset,\ T \neq \emptyset,\ L \subseteq N \times T \times N,$$

$$P\colon N \rightarrow C, \text{ that is } \forall c \in C,\ \exists u \in N \text{ such that } P(u) = c,$$

$$\forall n \in N,\ s \in S \text{ such that } n.type = s.$$

As in many other systems [Shackelford et al. 1993; Conklin 1987], there are two basic link types: organizational (T_o) and referential (T_r). Based on the two types, links are partitioned into two sets: a set of organizational links L_o and a set of referential links L_r.

$$T_o, T_r \subseteq T, \quad T_o \cup T_r = T, \quad T_o \cap T_r = \emptyset,$$

$$L_o \subseteq N \times T_o \times N,$$

$$L_r \subseteq N \times T_r \times N.$$

$$L_o, L_r \subseteq L, \quad L_o \cup L_r = L, \quad L_o \cap L_r = \emptyset,$$

Consequently, the graph G can be divided into two subgraphs G_o and G_r. G_o is formed with the organizational links. G_r is formed with the referential links. G_o is a rooted ordered acyclic graph (DAG), among whose nodes only one node has no predecessor. This node is a "dummy node" (χ) (it is dummy, because it is invisible to users, see next section). There is another special node named "top root node" (ω) which is the only successor of the "dummy node." In G_o, all nodes except for the "dummy node" are accessible from the "top root node," and all nodes except for the two special nodes may have common predecessors or common successors; but no node can be a descendant or ascendant of itself.

$$G_o = (N, L_o, C, P, T_o, S), \text{ where}$$

$$\exists \chi \in N,\ \forall m \in N,\ \forall t \in T_o,\ \langle m, t, \chi \rangle \notin L_o,$$

$$\exists \omega \in N,\ \exists \tau \in T_o, \text{ such that } \langle \chi, \tau, \omega \rangle \in L_o,$$

$$\forall s, t \in N \text{ and } s \neq \chi,\ t \neq \tau,\ \langle s, t, \omega \rangle \notin L_o,$$

$$\forall u \in N, \text{ there does not exist a path from } u \text{ to } u,$$

$$\forall u \in N \text{ and } u \neq \chi, \text{ there exists a path from } \omega \text{ to } u,$$

$$\forall u \in N,\ \exists \langle s, t, u \rangle \in L_o, \text{ where } s \in N \text{ and } t \in T_o.$$

$$G_r = (N, L_r, C, P, T_r, S)$$

In addition to being partitioned into organizational and referential types, links can also be partitioned into many semantic types based on the application domain concepts.

$$T_i \subseteq T, \quad T = \bigcap_{i=1}^{m} T_i, \quad T_i \cap T_j = \emptyset, \quad \text{where } i \neq j, \text{ and } i,j = 1, \ldots m.$$

By combining the above two link type partitions, a more detailed link type partition is given:

$$\Psi_T = \Psi_{T_o} \cup \Psi_{T_r}, \text{ where}$$

$$\Psi_{T_o} = \{T_{o_i}, \text{ for } i = 1, \ldots, m\} \quad \text{and}$$

$$\Psi_{T_r} = \{T_{r_i}, \text{ for } i = 1, \ldots, m\}.$$

In turn, a link partition is derived:

$$\Psi_L = \Psi_{L_o} \cup \Psi_{L_r}, \text{ where}$$

$$\Psi_{L_o} = \{L_{o_i}, \text{ for } i = 1, \ldots, m\} \quad \text{and}$$

$$\Psi_{L_r} = \{L_{r_i}, \text{ for } i = 1, \ldots, m\}.$$

3.2.2 NodeWeb Composition Scheme. M_{nw} is a composition scheme which maps from a given node to a composite (a NodeWeb) of nodes and links. A NodeWeb (NW) consists of a rooted DAG as its backbone together with other links attached to the backbone. The backbone is denoted as $NW_1(d)$, which has a "root node" d, and includes all the nodes that are accessible along organizational links from the "root node." The reason to define a "dummy node" a "top root node," and a link between them in G is to have an initial graph which conforms to the model and from which other structures can be appended. As a consequence, every node other than the dummy node is in an equal condition to serve as a root node for a NodeWeb, and every node has at least one incoming organizational link. For instance, a NodeWeb rooted from the "top root node" will be the whole user accessible hyperspace; and the NodeWeb generated from a node at a lower level could be a document or a part of a document. By including links that have one or both end points in $NW_1(d)$ (see definitions of L_i, i = 2, 3, 4, 5), four variants of NodeWebs ($NW_i(d)$, i = 2, 3, 4, 5) can be defined.

$$M_{nw}\colon (d, i, G) \rightarrow NW_i(d) = (N', L_i, C', P, T, S), \text{ where}$$

$$d \in N', i \in \{1, 2, 3, 4, 5\}, N' \subseteq N, C' \subseteq C,$$

$$\forall i \in \{1, 2, 3, 4, 5\}, L_i \subseteq L, \text{ where}$$

$$L_1 \subseteq N' \times T_o \times N',$$

$$\forall u \in N', \text{ in } NW_1(d), \text{ there exists a path from } d \text{ to } u,$$

$$\exists s \in (N - N') \text{ and } t \in T_o \text{ such that } \langle s, t, d \rangle \in (L_o - L_1),$$

$$L_2 \subseteq L_1 \cup (N' \times T_r \times N'),\ L_1 \subseteq L_2,$$

$$L_3 \subseteq L_2 \cup (N' \times T_r \times (N - N')),\ L_2 \subseteq L_3,$$

$$L_4 \subseteq L_3 \cup ((N - N') \times T_r \times N'),\ L_3 \subseteq L_4,$$

$$L_5 \subseteq L_4 \cup ((N - N') \times T_o \times N'),\ L_4 \subseteq L_5.$$

In other words, L_1 contains the organization links whose end points are in NodeWeb $NW_1(d)$; L_2 contains L_1 and all the referential links whose both end points are in the NodeWeb; L_3 includes L_2 and all the referential links whose target node is outside the NodeWeb; L_4 contains L_3 and all the referential links whose source node is outside of the NodeWeb; and L_5 includes L_4 and all the organizational links whose source node is outside of the NodeWeb. The reason for using NodeWeb rather than subgraph is that a NodeWeb in G as defined above may not be a subgraph in G, since some endpoints (nodes) of links in the NodeWeb may not belong to the NodeWeb. These variants of NodeWebs provide specified context for a collection of nodes and links, and provide a basis for the copying and view generating functions to be defined in next sections.

3.3 Rules

R_s is a set of domain-specific linking rules. A linking rule specifies whether a link of a certain type is allowed between two typed nodes. A linking rule is defined, similarly to a link, as a triple of ⟨node type, link type, node type⟩. For example, the linking rule of ⟨issue, generate, position⟩ specifies that a "generate" typed link can only be linked from an "issue" typed node to a "position" typed node.

The rule set is defined as

$$R_s \subseteq \{\langle s.type, t, d.type \rangle | s.type, d.type \in S, \text{ and } t \in T\}.$$

Any link must conform to a semantic linking rule (the relational constraint):

$$\forall \langle s, t, d \rangle \in L,\ \exists \langle s.type, t, d.type \rangle \in R_s,$$

$$\text{where } s, d \in N, \text{ and } t \in T.$$

Any node other than χ must have at least one incoming organizational link in G_o (the structural constraint):

$$\forall d \in N \text{ and } d \neq \chi,\ \exists \langle s, t, d \rangle \in L, \text{ such that } \langle s.type, t, d.type \rangle \in R_s,$$

$$\text{where } s, d \in N, \text{ and } t \in T_o.$$

3.4 Operators

The operators of the model are defined in a way that the rules reflecting the structural and relational constraints are respected. The theoretical proof of this computation property of the model can be found in Wang [1995]. The following are the major manipulation operators and their definitions.

O is a set of operators that manipulate the graph G.

O = {create-node-and-O-link, create-O-link, modify-O-link,

delete-O-link, delete-nodeweb, create-R-link, delete-R-link,

copy-node$_s$, copy-node$_{ns}$, copy-nodeweb$_s$, copy-nodeweb$_{ns}$}

Let $G = (N, L, C, P, T, S)$ be the graph to be manipulated and

$G' = (N', L', C', P, T, S)$ be the graph after manipulation.

(1) A new node is created together with the creation of a new organizational link. Linking rules will be checked before the creation is granted.

create-node-and-O-link: $(\langle s, t, d\rangle, G) \rightarrow G'$, where

$s \in N$, $t \in T_o$, $d \notin N$, $c' \notin C$,

$\exists\langle s.type, t, d.type\rangle \in R_s$,

$N' = \{d\} \cup N$, $L' = \{\langle s, t, d\rangle\} \cup L$, $C' = \{c'\} \cup C$.

c' is an empty content automatically created together with the node d.

(2) An organizational link can be created between two existing nodes to share the structure rooted from the target node of the new link. Linking rules and loop detection will be checked before the creation is granted.

create-O-link: $(\langle s, t, d\rangle, G) \rightarrow G'$, where

$s, d \in N$, $t \in T_o$ and $\langle s, t, d\rangle \notin L_o$,

$\exists\langle s.type, t, d.type\rangle \in R_s$,

$d \notin \Gamma^{*-}(s)$ in G_o,

$L' = \{\langle s, t, d\rangle\} \cup L$.

(3) An organizational link can be deleted, and any node (together with the links attached to the node) that become inaccessible from ω in G_o after the deletion of the organizational link will also be deleted.

delete-O-link: $(\langle s, t, d\rangle, G) \rightarrow G'$, where

$s, d \in N, d \neq \omega, t \in T, \langle s, t, d\rangle \in L_o,$

$M_{nw}(d, 1, G) = NW''_1(d),$

$NW''_1(d) = \{N'', T, L''_o, C'', P\},$

$N' = N - \{n | n \in N'', \text{ and } \neg \exists u \in (N - N'') \text{ such that } n \in \Gamma^{*+}(u) \text{ in } G_o\},$

$C' = C - \{c | c \in C'', \text{ and } \neg \exists u \in (N - N'') \text{ such that } P(u) = c\},$

$L' = (L - \{\langle s, t, d\rangle\}) - \{\langle n, t', d'\rangle \text{ and } \langle s'', t'', n\rangle | n \in N'', t', t'' \in T,$

$s' \in N \text{ and } \neg \exists u \in (N - N'') \text{ such that } n \in \Gamma^{*+}(u) \text{ in } G_o\}.$

(4) A node (and the structure rooted from the node) can be "moved" by changing the source of the organizational link pointing to it (actually by deleting the old organizational link and then creating a new one). Linking rules and loop detection will be checked before the creation is granted.

modify-O-link: $(\langle s, t, d\rangle, s', G) \rightarrow G'$, where

$s, s', d \in N, t \in T_o, d \neq \omega, \text{ and } \langle s, t, d\rangle \in L_o, \langle s', t, d\rangle \notin L_o,$

$d \notin \Gamma^{*-}(s') \text{ in } G_o$(i.e., d is not converse-accessible from s' in G_o),

$\exists \langle s'.type, t, d.type\rangle \in R_s,$

$L' = (L - \{\langle s, t, d\rangle\}) \cup \{\langle s', t, d\rangle\}.$

(5) A NodeWeb can be deleted. The effort of this operator differs from the delete-O-link operator in that this operator physically deletes the NodeWeb (and consequently all appearances of the nodes and links of the NodeWeb), while the delete-O-link operator may remove only one appearance of the NodeWeb if there are more than one incoming organizational link pointing to the NodeWeb (see Section 3.5 (NodeWeb presentation)).

delete-nodeweb: $(d, G) \rightarrow G'$, where

$d \in N, d \neq \omega,$

$M_{nw}(d, 5, G) = NW_5(d),$

$NW_5(d) = (N'', L'', C'', P, T, S),$

$N' = N - N'',$

$L' = L - L'',$

$C' = C - (C'' - \{c | c \in C'' \text{ and } \exists n \in (N - N'') \text{ such that } p(n) = c\}).$

(6) A referential link can be created between two existing nodes as long as the linking conforms with a linking rule.

create-R-link: $(\langle s, t, d\rangle, G) \rightarrow G'$, where

$s, d \in N$, $t \in T$ and $\langle s, t, d\rangle \notin L_r$,

$\exists\langle s.type, t, d.type\rangle \in R_s$,

$L' = \{\langle s, t, d\rangle\} \cup L$.

(7) A referential link can be deleted.

delete-R-link: $(\langle s, t, d\rangle, G) \rightarrow G'$, where

$s \in N$, $t \in T$, $d \in N$ and $\langle s, t, d\rangle \in L_r$,

$L' = (L - \{\langle s, t, d\rangle\}$.

(8) A content-sharing copy of a node can be made to share the content of an existing node. In this operator, an organizational link is also created to link the copy from a given node. The subscript "s" refers to "sharing" which means that the change to the node content will be shared by the original node and its content-sharing copy.

copy-node$_s$: $(\langle s, t, d'\rangle, d, G) \rightarrow G'$, where

$s, d \in N$, $t \in T_o$ and $d' \notin N$,

$N' = N \cup \{d'\}$, where $P(d') = P(d)$,

$\exists\langle s.type, t, d'.type\rangle \in R_s$,

$L' = L \cup \{\langle s, t, d'\rangle\}$,

$C' = C$.

(9) A copy of a node can be made that holds a duplicated content (hard copy) of the original node. An organizational link is created to link the copy from a given node. The "ns" refers to "nonsharing," which means that the content change of the copy and the original node are not shared. The value of c.blob is the blob of information corresponding to the content identifier c.

copy-node$_{ns}$: $(\langle s, t, d'\rangle, d, G) \rightarrow G'$, where

$s, d \in N$, $t \in T$, $d' \notin N$,

$N' = N \cup \{d'\}$, where $P(d') \neq P(d)$ and $P(d').blob = P(d).blob$,

$\exists\langle s'.type, t, d.type\rangle \in R_s$,

$L' = L \cup \{\langle s', t, d' \rangle\}$,

$C' = C \cup \{c'\}$, where $c' = P(d')$.

(10) A copy of a nodeweb can be made to share the contents of the original nodeweb. An organizational link is created to link the copy from a given node.

copy.nodeweb$_s$: $(\langle s, t, d' \rangle, i, d, G) \rightarrow G'$, where

$s, d \in N, d' \notin N, t \in T_o$,

$M_{nw}(d, i, G) = NW_i(d)$,

$i \in \{1, 2, 3, 4, 5\}$,

$NW_i(d) = (N'', L'', C'', P, T, S)$,

$NW'_i(d) = (N''', L''', C''', P, T, S)$,

$N''' = \{n''' | n''' \notin N, n'' \in N'', P(n''') = P(n'')\}$,

$L''' = \{\langle n''', t', d'' \rangle, \langle s', t'', n''' \rangle | n''' \in N''', n'' \in N'', p(n''') = p(n''),$

and $\langle n'', t', d'' \rangle, \langle s', t'', n'' \rangle \in L''\}$,

$N' = N \cup N'''$,

$\exists \langle s.\text{type}, t, d'.type \rangle \in R_s$,

$L' = L \cup L''' \cup \{\langle s, t, d' \rangle\}, P(d') = P(d)$,

$C' = C$.

(11) A copy of a nodeweb can be made that holds duplicated contents of the original nodeweb. An organizational link is created to link the copy from a given node.

copy-nodeweb$_{ns}$: $(\langle s, t, d' \rangle, i, d, G) \rightarrow G'$, where

$s, d \in N, d' \notin N, t \in T_o$,

$M_{nw}(d, i, G) = NW_i(d)$,

$i \in \{1, 2, 3, 4, 5\}$,

$NW_i(d) = (N'', L'', C'', P, T, S)$,

$NW'_i(d) = (N''', L''', C''', P, T, S)$,

$$C''' = \{c'''|c'' \in C'',\ c'''.blob = c''.blob\},$$

$$N''' = \{n'''|n''' \notin N,\ n'' \in N'',\ P(n''').blob = P(n'').blob\},$$

$$L''' = \{\langle n''',\ t',\ d''\rangle,\ \langle s',\ t'',\ n'''\rangle|n''' \in N''',\ n'' \in N'',$$

$$P(n''').blob = P(n'').blob, \text{ and } \langle n'',\ t',\ d''\rangle,\ \langle s',\ t'',\ n''\rangle \in L''\},$$

$$N' = N \cup N''',$$

$$\exists\langle s.type,\ t,\ d'.type\rangle \in R_s,$$

$$L' = L \cup L''' \cup \{\langle s,\ t,\ d'\rangle\},\ P(d').blob = P(d).blob,$$

$$C' = C \cup C'''.$$

3.5 Presentation

Since the whole graph of a structured hypertext is normally too large to be displayed on the limited space of the user interface, a method to display portions of the graph and to browse the graph is needed. Also, structural clues are needed to help people avoid getting lost, especially in the case that only portions of the whole graph are presented to them.

A fisheye view shows great detail for those parts of the information that are close to a user's current location of interest and gradually diminishing amounts of detail for those parts that are progressively farther away. The use of a fisheye view therefore requires two properties of the information space: it should be possible to estimate the distance between a given location and a user's current focus of interest, and it should be possible to display the information at several levels of detail [Nielsen 1990]. Both these conditions can be met with the structured hypertext. This is achieved by mapping the DAG skeleton of the structured hypertext into a virtual spanning tree to provide a hierarchical overview. In order to amplify the local view associated with the current focus of the hierarchical overview, a mapping from a single node to a spider (a node plus all its links) is defined.

In the following subsections, the hierarchical overview and the local view are defined. These two complementary structures are produced with functions of f_h and f_s. A third function f_t produces a strict tree rooted from a given node, since sometimes the duplication of contents is not desired (for example, when a paper copy of a document is to be produced). Based on the structures produced by these functions, the fisheye model is defined.

3.5.1 *Hierarchy and Spider.* The function f_h maps $NW_1(d)$ to a virtual tree. The generated tree is said to be virtual, since it may contain duplicated nodes or hierarchies (see Figure 2). Figure 2(1) presents a skeleton of $NW_1(A)$ (i.e., a root ordered DAG). Figure 2(2) is its hierarchical view generated by f_h. View1 and view2 share a subtree consisting of nodes C, D, and E. To generate a hierarchical view, the function f_h unfolds all the nodes that are accessible from a given node. The order of the

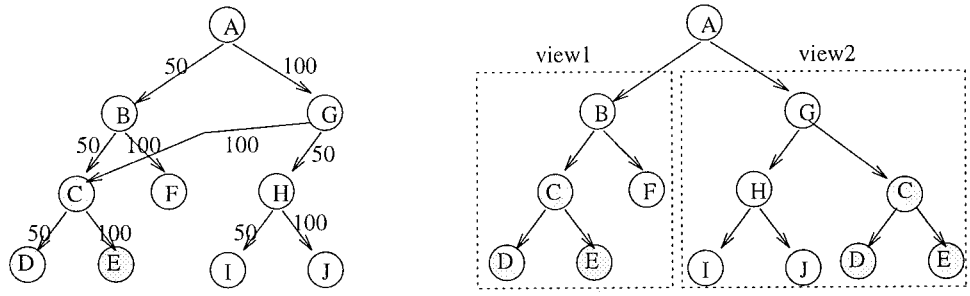

Fig. 2. DAG to virtual tree.

successors of a node is decided by the "order" attribute values of the outgoing links from the node (the smaller goes before the larger). The unfolding action ends only when leaf nodes (i.e., nodes whose out-degree is zero) or a given depth (a given control level) is reached. The rooted DAG structure of $NW_1(d)$ ensures that the unfolding action will terminate. The result of the unfolding is independent of the sequence in which nodes are unfolded, for instance, in a depth-first or in a breadth-first sequence. f_h is formally defined as

$$f_h\colon (NW_1(d),\ m) \rightarrow NW_{1_h}(d) = (N'',\ L_{o_h},\ C',\ P,\ T_o,\ S), \text{ where}$$

$$NW_1(d) = (N',\ L_1,\ C',\ P,\ T_o,\ S),$$

$NW_{1_h}(d)$ is an ordered (virtual) tree rooted from node d,

m specifies a control level or depth of the virtual tree,

$$N'' = N' \times Q,$$

$$L_{1_h} = \{\langle (s,\ q),\ t,\ (d,\ p)\rangle \mid \langle s,\ t,\ p\rangle \in L_1,\ p \text{ and } q \in Q\}.$$

Q is a set of unique labels which indicate the position of a node in the tree.

The depth-first traversal function f_t maps $NW_1(d)$ to a spanning tree [Carre 1979] in which each node appears at most once:

$$f_t\colon NW_1(d) \rightarrow NW_{1_t}(d) = (N',\ L_{1_t},\ C',\ P,\ T_o,\ S), \text{ where}$$

$$NW_1(d) = (N',\ L_1,\ C',\ P,\ T_o,\ S),$$

$NW_{1_t}(d)$ is an ordered spanning tree rooted from node d,

$$L_{1_t} \subseteq L_1.$$

f_s generates a local structure around a given node. This structure includes the node and all its incoming links and outgoing links. The

structure is similar to a spider in shape.

$$f_s\colon (d, G) \rightarrow NW_s(d) = (N''', L''', C, P, T, S), \text{ where}$$

$$G = (N, L, C, P, T, S),$$

$$d \in N,$$

$$N''' = \{d\},$$

$$L''' \subseteq (N''' \times T \times (N - N''')) \cup ((N - N''') \times T \times N''') \cap L.$$

3.5.2 Fisheye View Model. In the fisheye view model a *Degree of Interest* (DOI) function is used to determine whether or not a particular node should be shown in the current fisheye view [Furnas 1986]. One method is to divide the DOI into two components, one representing a priori interest (API) of a node, the other representing the distance $D(x, y)$ of node y from the current viewpoint x:

$$DOI(x, y) = API(y) - D(x, y)$$

$D(x, y)$ can be defined as the shortest path between the two nodes and API(y) = −D(root, y), which reflects the common sense that a node at a higher level in the hierarchy attracts higher interest [Furnas 1986].

The above definition of DOI applies to the rooted DAG (i.e., G_o) of the structured hypertext. However, since in G_o a node can be a descendant of more than one node (i.e., a node can belong to more than one document or perspective), it is reasonable to define the API of the node relative to the concerned document or perspective. On the other hand, when a view becomes too large to handle, the ability to zoom in and zoom out is desirable. This can be achieved by moving up and down the hierarchical skeleton through changing the "currentRoot." As G_o is presented to users as NW_{1_h}, the DOI is defined on the hierarchical structure of NW_{1_h}. Because NW_{1_h} is an ordered hierarchy, another factor affecting the API of a node is its order among its siblings. Therefore the DOI for the presentation of the hierarchical view (NW_{1_h}) of the structured hypertext can be defined as

$$DOI(x, y) = -D(currentRoot, y) - M * Order(y) - D(x, y),$$

$$\text{where } y \in \Gamma^{*+}(currentRoot).$$

The "order" is given a constant weight M (which is a fraction), so as to keep the "order" as a smaller factor than the distance.

In using the fisheye view model to reduce disorientation in hypertext, one suggestion is to designate certain nodes in the hypertext as landmarks, because of their familiarity, memorability, or salience. These would give users the chance to orient themselves around the objects that have known locations. The folding-unfolding functions upon NW_{1_h} can provide a fisheye

view with landmarks. Users can find what they want by unfolding to certain levels. Those levels left folded are landmarks telling them where they are.

Based on the above fisheye view formula, a set of operators are defined upon the multihierarchical structure NW_{1_h}. Three of them are defined for changing the current root:

—*Root* resets the "top root node" as the current root,
—*Top* sets the current highlighted node as the current root so as to zoom into a hierarchy of the current concern, and
—*Upper* sets a predecessor of the current root as the new current root so as to zoom out of a hierarchy in the structured hypertext.

Two unfolding functions allow users to look into different levels of detail (by setting $D(x, y)$ to be the levels of detail asked by users):

—*Unfold Node* function details the current viewpoint one level deep (i.e., display all successors of a node at the viewpoint, in this case $D(x, y) = 1$.
—*Unfold Document* displays nodes at all levels of the document (i.e., display all nodes accessible from the viewpoint, in this case $D(x, y) \geq 1$). Users can also ask to unfold a document to a given level (in this case $1 \leq D(x, y) \leq$ the given level).

The *Folding Node* function hides the details (hierarchy) of a node. The *Next* function navigates NW_{1_h} by taking all the three factors of the DOI into account (and the navigation actually is in the depth-first order on the weighted tree structure).

The *Display Node Content* function provides a means to review the detailed contents of a document. The *Display Node Content* function is also automatically activated by the *Next* function to allow the sequential reading of the logically structured document.

Users' interests may depend on many factors. An additional factor is added in the DOI formula to reflect a user's *current interest* (CI):

$$DOI(x, y) = CI(y) + [-D(currentRoot, y) - M * Order(y)] - D(x, y),$$

where $y \in \Gamma^{*+}(currentRoot)$

The current interest CI(y) can be specified by selecting and assigning some attribute values in the searching specification dialogue. The search results can be integrated with the folding/unfolding fisheye view by annotating the node names with the number of "hits," which can tell users not just *where* there is something of interest but also *how much* there is [Nielsen 1990]. After a search is performed, there will be two numbers associated with each node u. One is the number of the "hits" to the node itself, denoted by F(u). The other is the sum of the number of "hits" in the

hierarchy rooted from the node u, denoted by $F_h(u)$.

$$F_h(u) = \sum_{i=1}^{n} F(v_i), \quad \text{where } \{v_1, v_2, \ldots, v_n\} = \Gamma^{*+}(u).$$

In this way, a combination of searching and browsing can be supported, through which the browsing space can be reduced to a range that is relevant to the users' *current interest*.

Some of a user's *current interest* can be quantified, such as, the occurrence of a word or a string in a node, the times that a node was read by others, the number of links attached to a node, and the date of creation or modification. Some of a user's *current interest* may not have a direct numerical expression, such as authors and updaters. The search results can be provided in two ways: one is to distribute the numerical and symbolic information on the folding/unfolding hierarchical view and let users interpret them based on the *current interest* in their minds. The other is to produce a filtered view in which the "unhits" are removed, and the original hierarchical relation among the "hits" are respected. The filter function f'_h is defined as

$f'_h: NW_{1_h}(u) \rightarrow NW'_{1_h}(u)$, where

$NW_{1_h}(u) = (N''', L''', C''', P, T, S)$,

$NW'_{1_h}(u) = (N'''', L'''', C'''' P, T, S)$,

$N'''' = \{v | v \in N'''$, and $F_h(v) > 0\}$,

$L'''' = \{\langle u, t, v\rangle | u, v \in N'''', t \in T_o$, and $\langle u, t, v\rangle \in L'''$; or $\exists v_1, v_2, \ldots, v_r \in N'''$,

such that $u, v_1, v_2, \ldots, v_r, v$ is a shortest path in $NW_{1_h}(u)\}$.

$NW_{1_h}(u)$ is the original hierarchy rooted from node u, and $NW'_{1_h}(u)$ is the filtered hierarchy of $NW_{1_h}(u)$.

For example, given three nodes with the relation of grandfather, father, and son, if a search is performed from the grandfather node and the father node does not satisfy the search criteria, then the father node is filtered out, and in the new virtual tree, the son is promoted to take the father's position. The filtered view is "read only." When it is copied into a stable (static) document component, users can specify an organizational link type for this hierarchical structure. Therefore, the integrity of the structured hypertext model will not be violated by incorporating the filtered view function.

Similarly, a filtered spider view is defined as a filter function F'_s, which filters out certain typed links from $NW_s(d)$ that is generated by F_s.

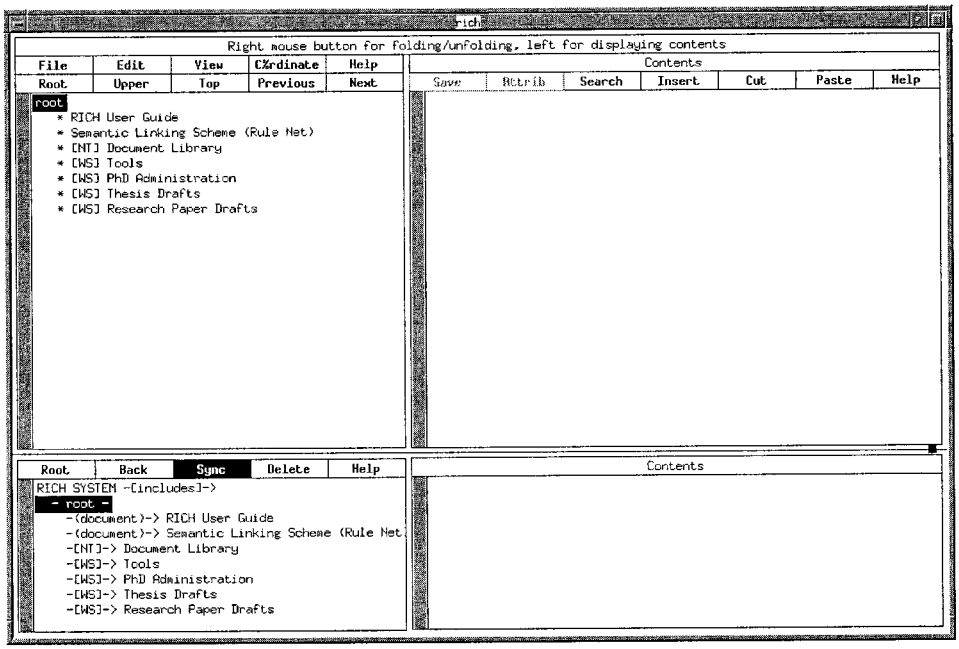

Fig. 3. RICH user interface for a structured document.

4. THE RICH SYSTEM AND ITS APPLICATION EXAMPLES

The formal model provides a solid basis for the design and implementation of the RICH system. The system runs on UNIX workstations. Its interface is built on X11 Athena Widget Sets, and its database system is built on a Btree package developed in the POSTGRES project at the University of California, Berkeley [Stonebrake 1990]. In its database system, in addition to databases for nodes and node contents, there is a link database together with two Btree indices for accessing all links coming from a node and all links going into a node. Because links in RICH are separate from information objects (node contents), the RICH system can construct hierarchical views and network views without the need to access nodes and their contents. Since the hierarchical view defined in the RICH model can be unfolded to deep levels and the hierarchical view may also include duplicated subhierarchies, memory space handling becomes very critical. The RICH approach to this problem is based on a method which can support quick operations with arbitrarily sized databases based on reading or updating from the database only what is displayed (unfolded) on the user interface (details of this can be found in Wang [1995]). This section describes the RICH system and some examples of its use to show how the formal model is matched to the system and how the reuse processes are supported with the system.

Figure 3 presents the implemented user interface. The separation of contents and structures, and the categories of organizational links and referential links are reflected on the user interface. And both the structures

and the contents can be seen simultaneously on the user interface. At the top of the interface is an information bar, which provides users with context-sensitive help, warning, and notification. Whenever a user positions the mouse over a certain area on the user interface, the relevant help information will be displayed in the information bar. Under the information bar there are some menus/buttons and four window areas that are divided into two parts. The upper part is a hierarchical browser for structured materials. On its left-hand side is a hierarchical overview (generated by f_h), and on its right-hand side is the content of a selected node (current node) in the hierarchical overview. The lower part is the network browser for the reference materials that are associated with the above logically structured materials. On its left-hand side is a "spider" structure (generated by f_s), and on its right-hand side is the content of a selected node in the "spider."

In the hierarchical browser, the right-hand-side mouse button is used for node folding/unfolding, and the left mouse button is used for displaying the content of a node. The buttons of Root, Upper, Top, Prev, and Next as defined in the formal presentation model allow the user to move up and down the hierarchical backbone of the structured hypertext. The semantic net manipulation operators defined in the formal data model can be activated from the Edit menu. The semantic net presentation operators defined in the formal presentation model can be activated from the View menu. The document import and export utilities that are based on the NodeWeb composition scheme can be activated from the File menu. The editor key bindings and buttons in the content area are patterned after those in Emacs.

In the network browser, the right-hand-side mouse button is used for changing the current focus of a spider view, and the left mouse button is used for displaying the content of a node. The Delete button corresponds to the delete referential link operator defined in the formal data model. The network browser can be used independently, or set to "synchronize" with the hierarchical browser (by toggling the Sync button), so that when the current node in the hierarchical view browser changes, the change of the spider view in the semantic net view browser follows.

In the following subsections, some application examples of the system are provided. These examples demonstrate how the system can be used for supporting the life cycle of document reuse processes, i.e., the document creation, management, retrieval, and reorganization processes.

4.1 Document Creation

The goal of the document creation process is to create the schemata of information (domain information organization structures) and use that schemata knowledge to develop document components (instances of the schemata). In the RICH system, a document schema is represented by a "rule net" (a type graph), whose nodes represent node types and whose links represent allowable link types between the node types. As shown in

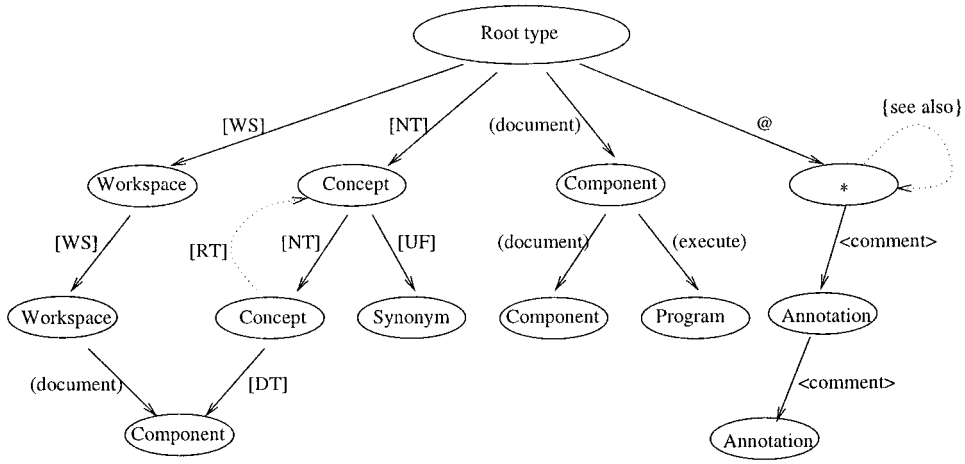

Fig. 4. Rule net.

Figure 4, rule links are defined in the context of a NodeWeb structure. The rules for relational constraints (represented with rule links) together with structural constraints built in the model constitute a definition of a domain model (which is similar to document type definition in SGML terms).

The rule net in Figure 4 is defined for the domain model that is briefly described in the domain model section of this article. In Figure 4, solid arrows represent organizational links, and dotted arrows represent referential links. The "*" (asterisk) is a wildcard for any node types, and the "@" (at) indicates that the link is not a rule link. Link types are represented with different brackets, i.e., [ ], ( ), ⟨ ⟩, and { } for thesaurus, document, annotation, and referential link type, respectively. The words in the brackets are link names, which are subtypes of the four general link types for thesaurus, document, annotation, and referential links. For instance, the linking rule ⟨Component, document, Component⟩ means a "document" type link can used between "Component" type nodes, and as this rule link is in organizational category, along links of the type no cycle is allowed. The content of a node in the rule net is the help message for the node type. This message will be activated when a linking rule related to the node type is violated.

The rule net is created in the same way that other hypertext structures are created. The differences is that its creation uses three variant versions of the create-node-and-O-link, create-O-link, and create-R-link functions. In these variant functions the checking for semantic linking rules is switched off, and in addition to their normal inputs, they also ask for a new type for the link and/or node to be created.

The structured hypertext model advocates (implies) a structured approach to construct structured hypertexts. One indication of this approach is reflected in the creation operator of new nodes. A new node has to be created together with an organizational link. The link type assignment in this operation can be seen as an indexing action for the new node (i.e., to

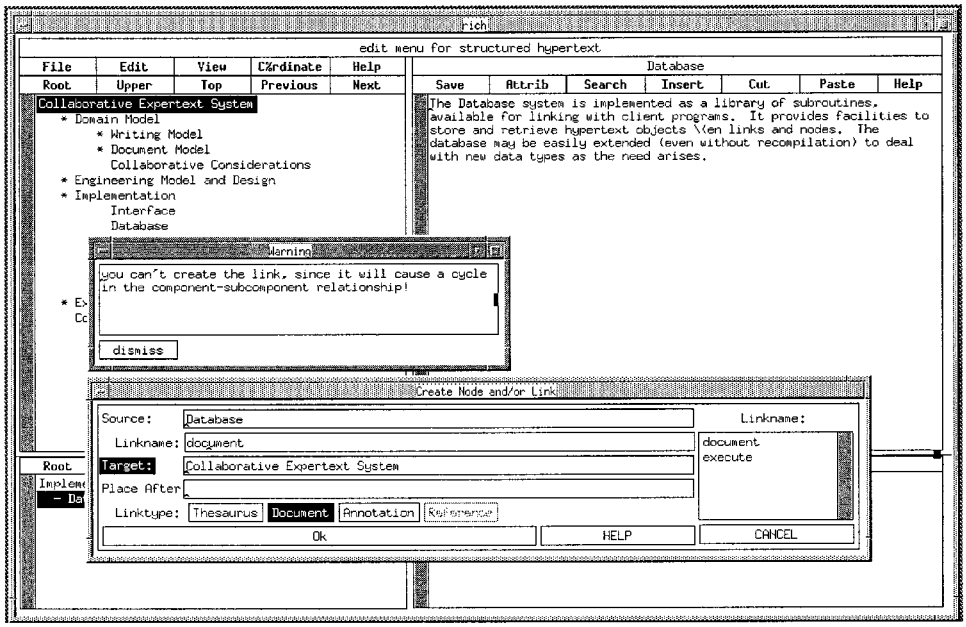

Fig. 5. Creating a link between two existing nodes. To create a link from the current node (the source node) to an existing node (the target node), users can select the target node in the hierarchical browser by first clicking on the "Target" area within the link creation dialog box, and then clicking on a target node name (the heading) in the hierarchical browser.

classify it under a certain hierarchy). When creating a new node or creating a link between two existing nodes, the structural constraints will be checked and then the relational constraints (domain linking rules) will be checked. For many cases, it is possible to prevent users from violating the rules concerning link types by providing a list of allowed links according to the source node type of the link to be created. There are also cases in which a violation cannot be foreseen. If a violation is detected, then a warning and explanation message will be displayed. As shown in Figure 5, a user intends to create a document-type organizational link from the node "Database" to the node "Collaborative Expertext System." As the node "Collaborative Expertext System" is an ancestor of the node "Database," a potential loop is detected. As a consequence the link creation is aborted, and a warning and explanation are given.

4.2 Document Management

Figure 6 shows how thesauri, documents, and annotations are presented in the RICH system. Link types are represented with different brackets, i.e., [ ], ( ), ⟨ ⟩, and { } for thesaurus, document, annotation, and referential link type, respectively. The words in the brackets are link names, which are user-defined subtypes of the four general link types. In the hierarchical browser (the upper part of Figure 6), a link is indicated by indentation and the brackets for document type are omitted. Only organizational links are

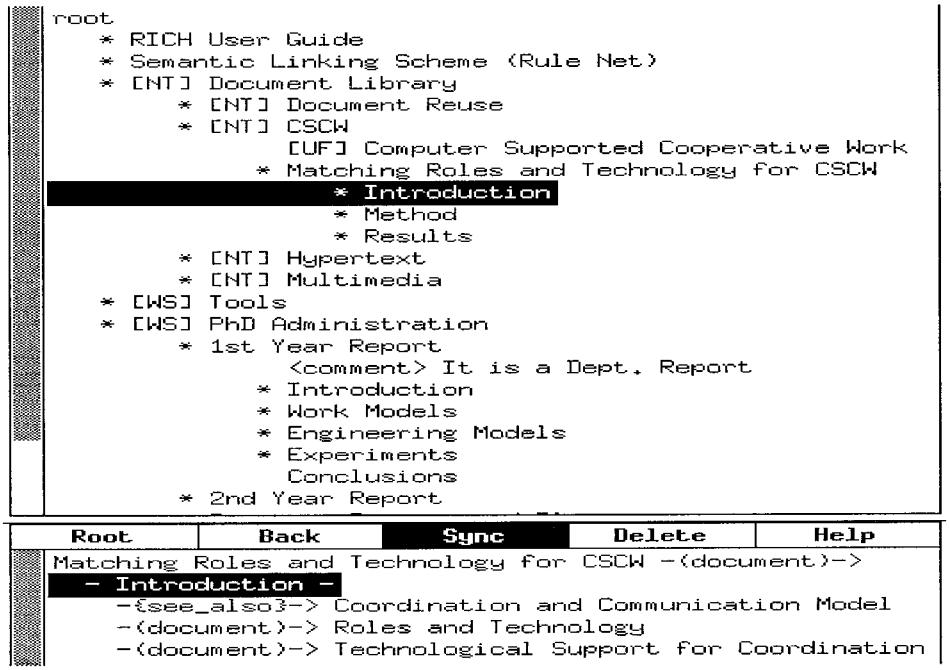

Fig. 6. Hyperspace organization.

visible in the hierarchical browser. In the network browser, all link types (or selected types of links) are presented and the direction of a link is indicated with an arrow line. As shown in Figure 6, the thesaurus of a document library is divided into four high-level topics: Document reuse, CSCW, Hypertext, and Multimedia. "The 1st Year Report" node has a comment of "It is a Dept. Report." The "Introduction of the The 1st Year Report" has a referential link to "Coordination and Communication Model." This referential link cannot be seen in the hierarchical view window, but can be seen in the spider view window.

The RICH NodeWeb composition schema provides a basis for implementing versioning, concurrency control, and access control upon a document composite. Currently, a simple user-controlled versioning mechanism is used. This versioning mechanism is implemented using the NodeWeb copying mechanisms. Versions of documents are made by explicit user operations upon the document composites.

4.3 Document Retrieval

The ability to browse is generally regarded as one of the most important reasons for using hypertext, however, searching facilities should also be supported in modern hypertext environments [Halasz 1988]. The hierarchical backbone of structured hypertext provides a structural clue to combining browsing with searching. By combining browsing with searching, the

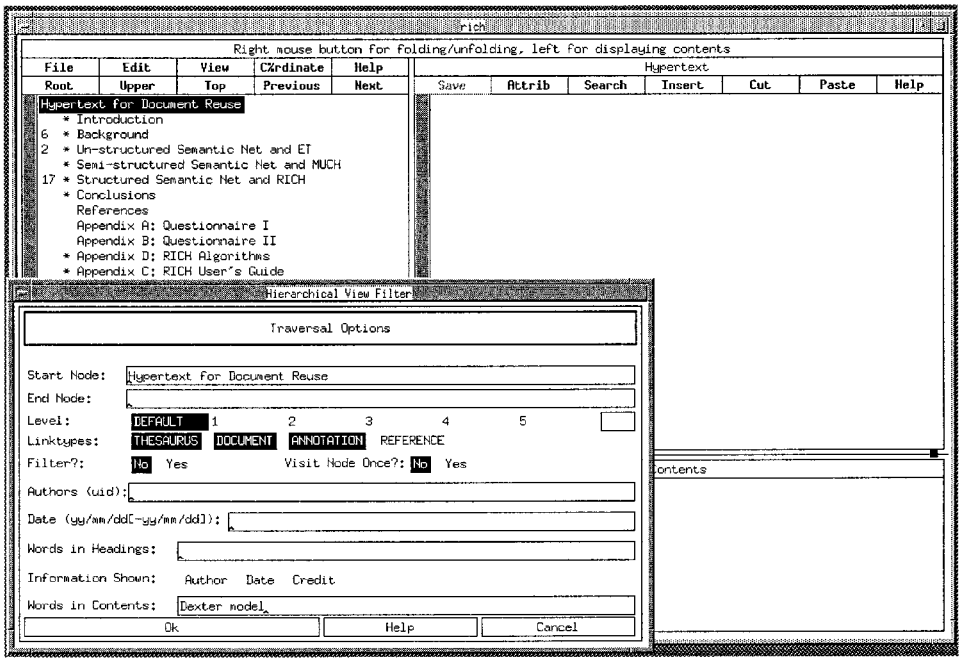

Fig. 7. Searching and indicating nodes hit. The document shown in the hierarchical browser is a draft of the thesis. By doing a free-text search, one can see that in the "Background" chapter, the term "Dexter model" has occurred six times.

browsing space can be reduced to a subnet as defined in the f'_h function in the formal presentation model.

In Figure 7, the traversal option dialogue box allows users to specify their current interest. The specification can be any combination of the criteria set in the dialogue box. A user is searching for the nodes whose contents relate to "Dexter model." The "Filter" attribute is set to "No," so no nodes will be removed by filtering. The searching results are "indicated" (with numbers of "hits") in the original context of the document. Indicating nodes hit in their original context may help users understand the relation among the found nodes. In RICH, similar to SuperBook [Egan et al. 1989], when a node is folded, the number at its side is the sum of "hits" in the hierarchies rooted from the node. When a hit node is unfolded, the number at its side is the number of "hits" to the node itself. By unfolding the hit node, the number may "move" or "reduce" (if there are other hits under the hierarchy). The numerical distribution on the outline can guide users to find the nodes they want by unfolding the numbered nodes with the hierarchical browser (see Figure 8).

4.4 Document Reorganization

The powerful feature of structured hypertext lies in its particular support for creating multihierarchies, which are the bases of multiple classification

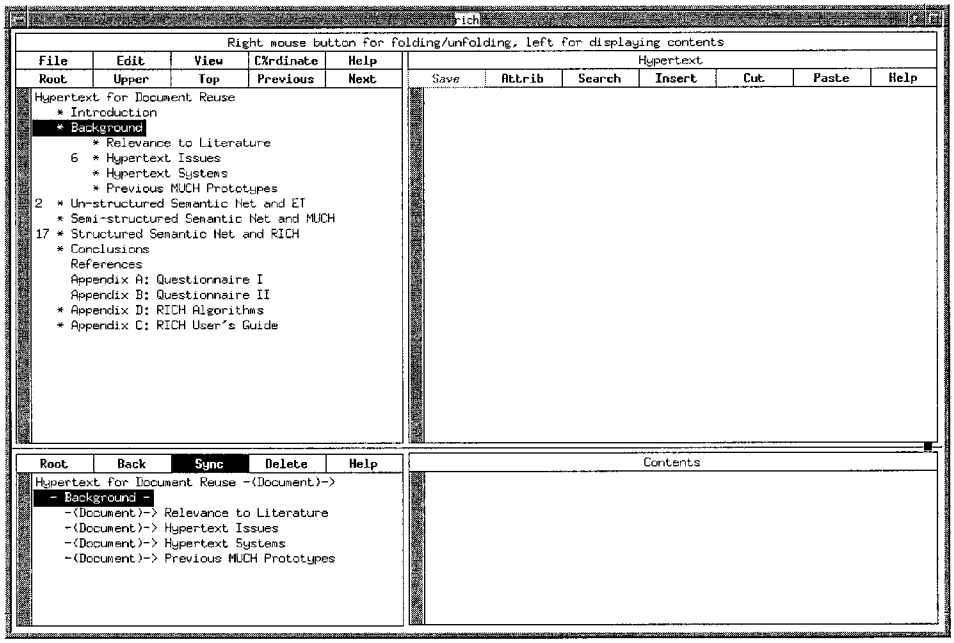

Fig. 8. Browsing hit nodes in original context. By unfolding the node "Background," one can see that the number "moves" to "Hypertext Issues."

schemes of a document library and multiple logical structures of a document. The multiple classification schemes provide users with alternative taxonomic views of the domain and thus provide alternative browsing and querying strategies for locating documents. The multiple logical structures provide alternative hierarchical views of a document and thus help users to understand the document from different perspectives. The RICH copying functions go beyond duplicating a set of linked nodes and contents. The novelty lies in their support for creating various multihierarchies that allow a mixture of sharing and nonsharing changes on content and structure.

A "copy" that shares changes on both content and structure of a document can be made with an organizational link creation function (create-O-link). The new organizational link goes from where the copy is supposed to be placed to the root node of the document to be copied. The shared structure in this situation includes all the nodes and links of the document. Although in the database, the document is not duplicated, in the hierarchical-view browser the document will be duplicated. This kind of copying can be used to create multiple classification structures of the document library and alternative views on shared document components.

A copy that shares changes on contents, but does not share changes on structure can be supported by the content-sharing copying function (copy-nodeweb$_s$). This kind of copying can create different structures (multiple high-level views or perspectives) on the same set of contents. This can be achieved by doing the copy first, then renaming the nodes and reorganizing

links in the copy into a new structure, and finally, if needed, modifying the shared contents from multiple perspectives to make them coherent to all these perspectives.

The copy that does not share changes on contents and structure can be made with the nonsharing copying function (copy-nodeweb$_{ns}$). This copying function is used when many content and structure changes are needed for integrating the copied components into a new document. The changes made in the copy will not affect the original document. A simple versioning mechanism of the RICH system is also implemented with this copying function. In addition to copying existing documents, the copying functions can also "copy" filtered views into a new document.

In practice, people may often need to use a combination of the above-described mechanisms. For example, in order to prepare a department prospectus more efficiently, a reusable component library can be built which contains a set of reusable document components, such as department descriptions, staff curriculum vitae, course descriptions, course programs, and equipment descriptions. The document components are continually being updated to reflect the latest changes. The department prospectus can be created with a combination of the above-described mechanisms. For instance, a course program can be prepared by using the content-sharing copying function to copy a similar course program, and then modify it into a new course program by renaming some courses, changing the order of some courses, removing some courses, and copying some courses from other course programs. In this case, the content changes made on the course descriptions in the library are reflected in the new course program, while the local structural changes made on the new course program will not affect the reusable components in the library. Suppose that the curriculum vitae of staff in the reusable component library are divided into several subsections, such as "personal data," "education," and "publication." In this case, the curriculum vitae entry in the prospectus can be created by simply creating organizational links to the curriculum vitaes in the library, so that both content and structure changes of curriculum vitae in the library will be reflected in the prospectus.

5. USER STUDIES

The RICH system and its underlying information reuse methodology have been used in two studies that compare and contrast the MUCH system with the RICH system. The first study was on creating a multiperspective hypertext document for managing the publications of a research group. The second study was on creating views of an existing document by computer science students.

5.1 Creating Hypertexts for Research Management

Three groups of two people each were asked to prepare a hypertext document as an organization manual for a research group. Two groups used the MUCH system, and one group used the RICH system. The require-

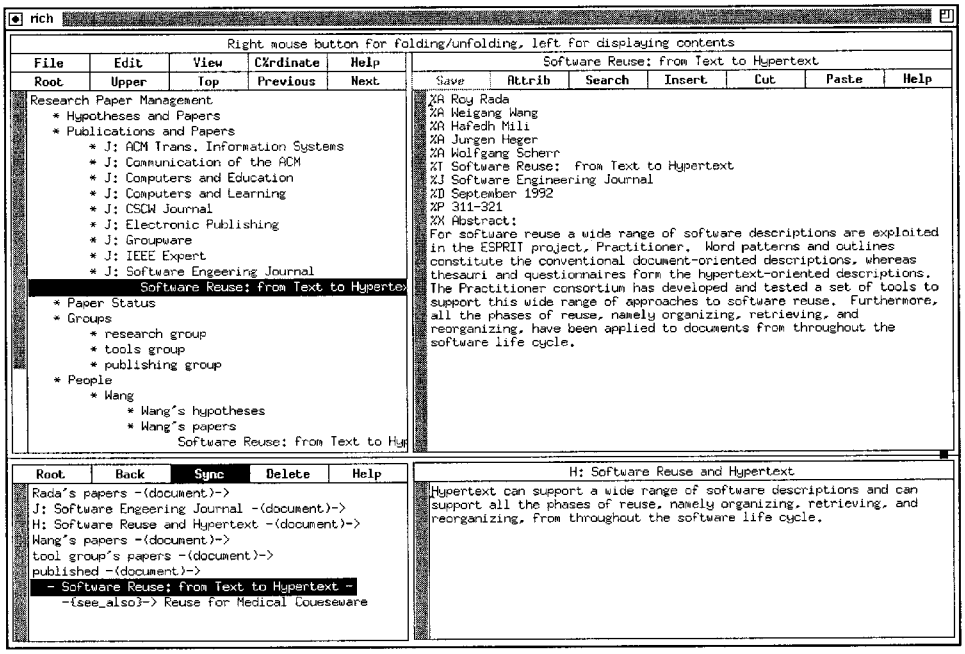

Fig. 9. A hypertext document for research management.

ments for the document were that it should be able to present users with multiple perspectives.

The two groups that used MUCH failed to produce a multiperspective document. They realized that creating a new link between two existing nodes or modifying a link might lead to radical consequences (different outlines or views). Since there is no constraint on linking, these problems were quite serious. The document they finally created had only one hierarchy.

The two people who used RICH successfully produced a multiperspective document (see Figure 9). With the hierarchical browser, some node in the document can appear in different hierarchies, for example the "Software Reuse" paper appears under both "Software Engineering Journal" and "Wang's papers." Users can unfold different headings at the top levels for different perspectives. From the semantic net browser, users can see that the "Software Reuse" paper has multiple incoming links, which tells users that the paper was coauthored by Rada and Wang, published in the *Software Engineering Journal*, and related to a hypothesis on software reuse and hypertext.

5.2 Creating Views on a Book

The MUCH system and the RICH system were also used in a five-week course on "interactive media" for computer science students. A document reuse task was assigned to the class. The aim of this exercise was to see

which view-creating method users would prefer. Each student used both the MUCH system and the RICH system, but the students were divided into two groups randomly. One group used first the MUCH system, and then the RICH system, while the other group used first the RICH system and then the MUCH system.

In comparing the view-creating methods underlying the two systems, 74% of the students that used MUCH first preferred the RICH method. In the group that used RICH first, 90% of the students preferred the RICH method. In their reports, several students commented that with the MUCH system the absence of some parts of the original document (caused by the creation of new links) was confusing. Students liked RICH for its context-sensitive help mechanism, interface layout, and immediate visibility of the views created.

6. RELATED MODELING ENGINEERING WORK

Several researchers have combined the features of hypertext and structured documents. The approach of Grif [Quint and Vatton 1992] is to add hypertext features into structured documents by integrating typed links into a structured document model, while the RICH approach is to incorporate logical structures into hypertext. The HTML documents used in the World Wide Web and the structured documents used in the Andrew multimedia editor [Grantham et al. 1987] are other examples of incorporating hypertext links into structured documents. Like in Grif, in Andrew and HTML documents, a node is usually the whole document that includes hard-coded tags for section headings. Comparing with the RICH approach, their drawback is the lack of granularity of information units and the separation between structure and content. The structures and constraints of the structured hypertext are very similar to a grammar or a SGML DTD for structured documents. The difference lies in the flexible handling of referential links and in the separation between content and structure. In our model, no tags on high-level logical structure are embedded in node contents. These tags (such as ⟨h1⟩ and ⟨h2⟩) are added only when a document is exported. This makes the levels of a node in the hierarchical structures relative to one another. Thus, these substructures can be freely moved across different levels of a hierarchical structure in the document reorganization processes without a need to change the tags in the contents.

The RICH structured-hypertext model is similar to the hypertext models of the DGS system [Shackelford et al. 1993] and Hyper-G (HyperWave) [Andrews et al. 1995]. The classification of organizational and referential links in RICH corresponds to the classification of structural and hyper-structural links in DGS. The DAG backbone together with referential links in RICH corresponds to the DAG structure of nested collections together with external hyperlinks in Hyper-G. However, there are also many differences between them. In DGS, subgraphs are embedded in node contents. Before opening a node, links within the node cannot be seen. In Hyper-G, documents or subcollections (of documents or clusters) in a collection

cannot be seen before the collection is unfolded. The node-nesting structure in DGS and the collection-nesting structure in Hyper-G are nonlinking composition structures. In RICH, all links are external links that are stored in a link database separated from node contents. High-level structures (hierarchical and network overviews) are constructed on-the-fly from the link database not necessarily to access node contents. Although it is possible for DGS and Hyper-G to use their external links for modeling hierarchical structures, they do not place an emphasis on external-link-based composites. In addition, the RICH model is not only a graph-based model, but also a semantic-net-based model. Nodes and links in RICH are semantically typed to model application domain concepts, and rules are defined to maintain the consistency of the semantic net.

The structured-hypertext model is based on the Dexter model [Halasz and Schwartz 1990; Gronbak and Trigg 1994]. What is added to it are the semantic data-modeling constructs: the typed links, typed nodes, and the NodeWebs with domain semantics. With the model, these constructs are recognizable not only by human users, but also by the hypertext system. The Dexter model requires the containment relationship in a composite component to be a DAG; in this work, a rooted ordered DAG (G_o) and its operators have been formally defined. The Dexter model intentionally leaves the structure within node contents (i.e., the within-component layer [Halasz and Schwartz 1990]) undefined, so is the RICH model. Also this work incorporates the relational constraints into the Dexter-based model to govern its semantic net construction. A very detailed comparison of the formal model with the Dexter model can be found in Wang [1995].

Unlike many other composition schemes which construct a composite by grouping links and/or nodes into a logical collection with a single identifier [Delisle and Schwartz 1987; Haan et al. 1992], the NodeWeb scheme constructs a composite from external links at run time. Since there is no need to assign a unique identifier to all the links or all the nodes in a NodeWeb, a NodeWeb can be easily linked with or contained within other NodeWebs. Moreover, in addition to using links to separate contexts, the NodeWeb can also separate contexts with nodes. For instance, when users want to reuse the content of a node but do not want some of the links from that node, they can use the content-sharing copy function to create a new node that shares the content of that node. In Intermedia [Haan et al. 1992], a user can only open one "web" at a time. In RICH, a user can open many NodeWebs at a time. Actually the whole hyperspace is a big NodeWeb whose nodes can be unfolded to see smaller NodeWebs.

A composite node is a real entity that captures the non-linking-based organization of information [Gronbaek and Trigg 1992]. The NodeWeb scheme can provide users with an illusion of a composite, which is not a composite node, but it is equivalent to the effect of a composite node. The methods of building hierarchies in Augment [Engelbart 1984] and KMS are similar to the NodeWeb scheme. The differences lie in that the hierarchies and multihierarchies in RICH are generated along organizational links, which are not necessarily tree links. The structure underlying the hierar-

chies is a "rooted ordered DAG." This structure can meet the needs for multihierarchical classifications of a large document collection and for multiple perspectives (or hierarchical views) of a document. In Bench-Capon and Dunne [1989], a DAG structure and a set of constraints are used to model electronic documents. In that model links are not typed and only represent the containment relationship. The noncontainment relationship is not captured as links. The formal hypertext model described in Tochtermann and Dittrich [1995] provides some formally defined structural concepts. However, the model does not deal with typed nodes and links, and it does not have mechanisms to define the structural and the relational constraints upon hypertexts. The uniqueness of the RICH approach from the above efforts is that a semantic data model has been developed, which handles not only the DAG formed by the organizational links, but also the referential links attached to the DAG.

SEPIA [Streitz et al. 1992], gIBIS [Conklin 1988], and PHIDIAS [McCall et al. 1990] support predefined types created by system developers. Aquanet [Marshall et al. 1991], OVAL [Malone et al. 1992], and MacWeb [Nanard and Nanard 1991] use certain formal representations to allow knowledge structures to be defined by schema designers. The approach presented in this article allows ordinary users to define domain information models by creating a rule net which is largely the same as an ordinary hypertext document.

The RICH copying functions are similar to the hypertext templates described in Catlin et al. [1991]. They allow duplicating prelinked documents in one step so as to facilitate the creation of consistent hypertext collections. In their system, the duplicated templates are hard copies. In the RICH system, by allowing a mixture of sharing and nonsharing of changes on content and structure, the various copying functions have gone beyond a means for duplicating hard copies into a method for creating multihierarchies.

The RICH user interface is derived from the "book" metaphor in SuperBook [Egan et al. 1989] and the "frame" in KMS. The RICH user interface presents users with the logically structured materials together with the referential materials that relate to the logically structured materials. This combination matches closely to the formal model of the structured hypertext and provides users with a simple metaphor of the structured hypertext: multihierarchy plus cross-references.

7. CONCLUSION

Unstructured or semistructured semantic nets have been widely used as a logical model for hypertext systems [McKnight et al. 1991]. Hypertext systems developed with the model give users ultimate authority to structure information. However, they are inadequate for composing stable documents and for maintaining semantic consistency. We propose a formal semantic data model for structured hypertext. It includes four components: a graph representation, a composition scheme, a set of rules, and a set of

operators. With the model, many classes of documents (such as thesauri, documents, and annotations) can be represented, and the structural and relational integrity of the underlying semantic net can be maintained. One application of the model is to support the creation and management of documents of a given class by multiple authors. Enforcement of the structural and the relational constraints would ensure that the document structure and semantics were maintained, whatever modifications were made. The semantic data model is designed to allow data to be modeled in a manner that is analogous to the user's view of the application. Users are free of the requirement to model a variety of different classes of documents with just one set of modeling constructs.

The RICH system has been implemented using the structured hypertext model. The system can define and enforce a set of domain-specific rules to maintain the integrity of the semantic net. It provides by default a primitive set of node types and link types for the general application domain concepts of thesauri, documents, and annotations. Users can use the predefined types or make subclasses of the provided types for their applications; for instance, they can make subclasses of thesaurus type for topic-oriented document classification, or make subclasses of the document class for specific document types. As the rule net is created in the same way as a hypertext document is authored, users have no need to learn complicated formal syntax (or formal languages). This may make the creation of domain information models easier for ordinary users. In addition, the system can make use of the schema knowledge derived from the rule net to help users using the predefined types. Some examples and user studies are given, which demonstrate that the system provides particular support for creating multihierarchies with the reuse of existing contents and structures. The experiences with the RICH system and its predecessor, the MUCH system, show that meaningful information structures can be reliably maintained only when the link and node typing information becomes machine recognizable and applicable knowledge.

Many efforts have been made to incorporate a richer structure into the World Wide Web. The World Wide Web Consortium has developed updated versions of HTML with typed links [W3C 1997]. Also many full-fledged hypermedia systems have built link services or information servers that add advanced hypertext features to the WWW, such as composites and external links. Some of the well-known systems are Hyper-G (HyperWave) [Andrews et al. 1995], Microcosm (Multicosm) [Carr et al. 1996], and DHF/WWW [Gronbak and Trigg 1996]. Hyper-G emphasizes hierarchical information organization structures. Microcosm advocates generic links and an open hypermedia architecture to connect various information objects (documents) in their original formats (and with their own editors). DHF/WWW tries to exploit a general model and framework to augment existing WWW with Dexter-based structuring constructs. Although many hypermedia systems, including all the above-mentioned systems, support typed links that are separate from information objects, they use external

links primarily for cross-reference relations or for constructing paths. The composition structures are usually created by means of nonlinking mechanisms, such as nested collections in Hyper-G [Andrews et al. 1995] and nested document parts (or subgraphs) in DGS [Shackelford et al. 1993]. Compliment to above approaches, the RICH system made a focus on external-link-based composition constructs.

Although in most cases, especially in a single system, nonlinking composition mechanisms are more efficient than external-link-based composition mechanisms [Carr et al. 1996], there are also many cases where external-link-based constructs are desired [Gronbak and Trigg 1996]. Compared to nonlinking mechanisms, external-link-based composites have many distinct properties. They are constructed on-the-fly, and so they are more dynamic. They offer users to see and trace not only outgoing links from a node, but also incoming links to a node. They provide descriptive link types and names to help users in their navigation. They lend themselves to reuse existing links, nodes, and contents to create multiple overlapping views (and paths) upon the same set of information objects. They allow users to create links from documents to which they have no write-permission or from documents that have no public link embedding interface. These properties are valuable in augmenting the embedded link (jump address) based hypertext in the current WWW.

Some methods for augmenting the WWW with external links have been developed and tested [Gronbak and Trigg 1996; Andrews et al. 1995]. One of the methods is to develop link services and Java applets that can run in any Java-enabled WWW browsers. Our next work is to exploit these methods so as to apply the RICH composition scheme and its user interface to the WWW. This will incorporate the RICH external-link-based constructs with the WWW embedded links and HTML-based constructs, and provide users with hierarchical, network overviews and other benefits of a well-organized structured hypertext.

REFERENCES

AFRATI, F. AND KOUTRAS, C. D. 1990. A hypertext model supporting query mechanisms. In *Proceedings of the ECHT'90 European Conference on Hypertext* (1990), 52–66.

AKSCYN, R., MCCRACKEN, D., AND YODER, E. 1988. KMS: A distributed hypermedia system for managing knowledge in organizations. *Commun. ACM, 31,* 7, 820–835.

ANDREWS, K., KAPPE, F., AND MAURER, H. 1995. Serving information to the web with hyper-G. *Comput. Netw. ISDN Syst. 27,* 6.

BENCH-CAPON, T. AND DUNNE, P. 1989. Some computational properties of a model for electronic documents. *Electr. Pub. Orig. Dissem. Des. 2,* 4, 231–256.

CARR, L., HILL, G., ROURE, D. D., HALL, W., AND DAVIS, H. 1996. Open information services. In *Proceedings of 5th International World Wide Web Conference.*

CARRE, B. 1979. *Graphs and Networks.* Clarendon Press, Oxford.

CATLIN, K. S., GARRETT, L. N., AND LAUNHARDT, J. A. 1991. Hypermedia templates: An author's tool. In *Proceedings of ACM Hypertext'91.* 147–160.

CHUA, T. S. AND LAI, E. P. M. 1991. Supporting composition in a hypermedia environment. *Hypermedia 3.3,* 207–238.

CONKLIN, J. 1987. Hypertext: An introduction and survey. *Computer 20,* 9, 17–41.

CONKLIN, J. 1988. M. begeman gibis: A hypertext tool for exploratory policy discussion. *ACM Trans. Office Inf. Syst. 6,* 4, 303–331.

CONKLIN, J. AND BEGEMAN, M. 1989. gibis: A tool for all reasons. *J. Am. Soc. Inf. Sci. 40,* 3, 200–213.

DECOUCHANT, D., QUINT, V., AND ROMERO, M. 1996. Structured and distributed cooperative editing in large scale network. In *Groupware and Authoring,* R. Rada Ed. Academic Press, London, 265–296.

DELISLE, N. AND SCHWARTZ, M. 1987. Contexts—A partitioning concept for hypertext. *ACM Trans. Office Inf. Syst. 5,* 2, 168–186.

EGAN, D. E., REMDE, J. R., GOMEZ, L. M., LANDAUER, T. K., EBERHARDT, J., AND LOCHBAUM, C. C. 1989. Formative design-evaluation of 'superbook'. *ACM Trans. Inf. Syst. 7,* 1, 30–57.

ENGELBART, D. 1984. Authorship provisions in augment. In *IEEE Compcon Conference.*

EVENSON, S., RHEINFRANK, J., AND WULFF, W. 1989. Towards a design language for representing hypermedia cues. In *ACM Hypertext'89 Proceedings.* 83–92.

FURNAS, G. 1986. Generalized fisheye views. In *CHI'86 Proceedings.* 16–23.

GARG, P. K. 1988. Abstraction mechanisms in hypertext. *Commun. ACM, 31,* 7, 862–870.

GATES, B. 1996. Keynote lecture presented at comdex'96. *Available via http://www.microsoft.com/corpinfo/bill-g/speeches/comdex96/speech.htm.*

GRANTHAM, D., ROBERTSON, J., SUBASIC, K., AND LANGSTON, D. 1987. *A Guide to Andrew.* Information Technology Center, Carnegie Mellon University, Pittsburgh, Pa.

GRONBAEK, K. AND TRIGG, R. 1992. Design issues for a dexter-based hypermedia system. In *Proceedings of the 4th ACM Conference on Hypertext.* 191–200.

GRONBAK, K. AND TRIGG, R. 1994. Design issues for a dexter-based hypermedia system. *Commun. ACM 37,* 2 (Feb.), 40–49.

GRONBAK, K. AND TRIGG, R. 1996. Towards a dexter-based model for open hypermedia: Unifying embedded references and link objects. In *Proceedings of ACM Hypertext'96.* 16–20.

HAAN, B. J., KAHN, P., RILEY, V. A., COOMBS, J. H., AND MEYROWITZ, N. K. 1992. IRIS hypermedia service. *Commun. ACM 35,* 1, 36–51.

HALASZ, F. 1988. Reflections on notecards: Seven issues for the next generation of hypermedia systems. *Commun. ACM 31,* 7, 836–855.

HALASZ, F. AND SCHWARTZ, M. 1990. The dexter hypertext reference model. In *Proceedings of the Hypertext Standardisation Workshop.* U.S. Government Printing Office, 95–134.

HALASZ, F. AND SCHWARTZ, M. 1994. The dexter hypertext reference model. *Commun. ACM 37,* 2, 30–39.

INCE, D. 1990. Set piece. In *System and Software Requirements Engineering*, R. H. Thayer and M. Dorfman Eds., IEEE Computer Press, 370–372.

KOO, R. 1989. A model for electronic documents. *ACM SIGOIS Bull. 10,* 1, 23–33.

LEHMANN, F. W. 1992. Semantic networks. In *Semantic Networks in Artificial Intelligence*, F. W. Lehmann, Ed., Pergamon Press Ltd, 1–50.

LESK, M. E. 1969. Word-word associations in document retrieval systems. *Am. Doc. 20,* 1, 27–38.

LEVY, D. M. 1994. Fixed or fluid? document stability and new media. In *Proceedings of the ACM European Conference on Hypertext Technology* (Edinburgh). 24–31.

LUNG, C. AND URBAN, J. E. 1993. Integration of domain analysis and analogical approach for software reuse. In *Proceedings of the 1993 ACM/SIGAPP Symposium on Applied Computing* (New York), E. Deaton, K. M. George, H. Bergel, and G. Hedrick Eds., ACM Press, 48–53.

MALONE, T., LAI, K., AND FRY, C. 1992. Experiments with oval: A radically tailorable tool for cooperative work. In *Proceedings of ACM CSCW'92* (Toronto, Canada, Oct. 31–Nov. 4). 289–297.

MARSHALL, C. C. AND IRISH, P. M. 1989. Guided tours and on-line presentations: How authors make existing hypertext intelligible for readers. In *ACM Hypertext'89 Proceedings.* 15–26.

MARSHALL, C. C., HALASZ, F., ROGERS, R. A., AND JANSSEN, W. C., JR. 1991. Aquanet: A hypertext tool to hold your knowledge in place. In *Proceedings of ACM Hypertext'91.*

McCall, R., Bennett, P., D'Oronzio, P., Ostwald, J., Shipman, F., and Wallace, N. 1990. PHIDIAS: Integrating CAD graphics into dynamic hypertext. In *Proceedings of the ECHT'90 European Conference on Hypertext.* 152–165.

McKnight, C., Dillon, A., and Richardson, J. 1991. *Hypertext in Context.*

Mili, H. and Rada, R. 1988. Merging thesauri: Principles and evaluation. *IEEE Trans. Patt. Anal. Mach. Intell. 10,* 2, 204–220.

Nanard, J. and Nanard, M. 1991. Using structured types to incorporate knowledge in hypertext. In *Proceedings of ACM Hypertext'91.* 329–343.

Nelson, T. H. 1995. The heart of connection: Hypermedia unified by transclusion. *Commun. ACM 38.*

Nielsen, J. 1990. *Hypertext and Hypermedia.* Academic Press, Inc., San Diego.

Parsaye, K., Chignell, M., Khoshafian, S., and Wong, H. 1989. *Intelligent Databases.* John Wiley and Sons, Inc.

Parunak, H. V. 1991. Don't link me in: Set based hypermedia for taxonomic reasoning. In *Proceedings of ACM Hypertext'91.* 233–242.

Potter, W. D. and Trueblood, R. P. 1988. Traditional, semantic, and hyper-semantic approaches to data modelling. *IEEE Comput. 21,* 6, 53–63.

Prieto-Diaz, R. and Freeman, P. 1987. Classifying software for reusability. *IEEE Softw. 4,* 1, 6–16.

Quint, V. and Vatton, I. 1992. Combining hypertext and structured document in grif. In *Proceedings of the 4th ACM Conference on Hypertext.* 23–32.

Rada, R. 1990. Hypertext writing and document reuse: the role of a semantic net. *Electr. Pub. 3,* 3, 3–13.

Rada, R. 1992. Converting a textbook to hypertext. *ACM Trans. Inf. Syst. 3,* 294–315.

Rada, R., Wang, W., and Birchall, A. 1992. An expertext system for collaborative authoring. *Exp. Syst. Appl. 5,* 3/4, 275–288.

Rada, R., Zeb, A., You, G., Michailidis, A., and Mhashi, M. 1991. Collaborative hypertext and the MUCH system. *J. Inf. Sci. Principles Pract. 17,* 191–196.

Schnase, J. L., Leggett, J. J., Hicks, D. L., and Szabo, R. L. 1993. Semantic data modelling of hypermedia associations. *ACM Trans. Inf. Syst. 11,* 1, 27–50.

Shackelford, D. E., Smith, J. B., and Smith, F. D. 1993. The architecture and implementation of a distributed hypermedia storage system. In *Proceedings of ACM Hypertext'93.* 1–13.

Soergel, D. 1974. *Indexing Languages and Thesauri: Construction and Maintenance.* Wiley, New York.

Spivey, J. M. 1989. *The Z Notation—A Reference Manual.* Prentice Hall.

Stonebraker, M., Rowe, L., and Hirohama, M. 1990. The implementation of POSTGRES. *IEEE Trans. Knowl. Data Eng. 2,* 1, 125–140.

Stotts, P. D. and Furuta, R. 1989. Programmable browsing semantics in trellis. In *Proceedings Hypertext '89.* 27–42.

Stotts, D., Dewan, P., Munson, J., and Navon, J. 1996. A three level binding for collaborative editing semantics. In *Groupware and Authoring*, R. Rada Ed. Academic Press, London, 297–326.

Stotts, P. D., Furuta, R., and Ruiz, J. C. 1992. Hyperdocuments as automata: Trace-based browsing property verification. In *Proceedings of the 4th ACM Conference on Hypertext.* 272–281.

Streitz, N., Haake, J., Hannemann, J., Lemke, A., Schuler, W., Schutt, H., and Thuring, M. 1992. SEPIA: A cooperative hypermedia authoring environment. In *Proceedings of ACM Hypertext'92.* 11–22.

Tochtermann, K. and Dittrich, G. 1995. Towards a family of formal models for hypermedia. In *HIM'95 Proceedings.* 77–91.

Tomek, I., Khan, S., Muldner, T., Nassar, M., Novak, G., and Proszynski, P. 1991. Hypermedia—introduction and survey. *J. Microcomput. Appl. 14,* 63–103.

Trigg, R. H. 1996. Hypermedia as integration: Recollections, reflections and exhortations. In *Keynote Address in Hypertext'96 Conference.* Xerox Palo Alto Research Center.

Ullman, J. 1983. *Principles of Database Systems.* Pitman Publishing Limited, London.

W3C. 1997. The World Wide Web Consortium activity on HTML. *Available via http://www.w3.org/MarkUp/*.

WANG, W. 1995. Semantic net based hypertext for authoring and reuse. *Ph.D. dissertation.* Department of Computer Science, Liverpool University.

WANG, W. AND RADA, R. 1995. Experiences with semantic net-based hypermedia. *Int. J. Hum.-Comput. Stud. 43,* 419–433.

ZELLWEGER, P. T. 1989. Scripted documents: A hypermedia path mechanism. In *ACM Hypertext'89 Proceedings.* 1–14.

Received August 1995; revised December 1996 and June 1997; accepted November 1997