Int. J. Human–Computer Studies (1997) **47,** 31–65

Fourth generation hypermedia: some missing links for the World Wide Web

Michael Bieber

New Jersey Center for Multimedia Research; National Center for Transportation and Industrial Productivity; Electronic Enterprise Engineering Program; Institute of Integrated Systems Research; New Jersey Institute of Technology, University Heights—Newark, NJ 07102, USA. email: bieber@cis.njit.edu

Fabio Vitali

Department of Computer Science, University of Bologna, Mura Anteo Zamboni 5, I-40121 Bologna BO, Italy. email: fabio@cs.unibo.it

Helen Ashman

Information Technology Division, Defence Science and Technology Organisation, P.O. Box 1500, Salisbury, S.A. 5108, Australia. email: helen.ashman@dsto.defence.gov.au

V. Balasubramanian

E-Papyrus, Inc., and Graduate School of Management, Rutgers University, Newark, NJ 07102, USA. email: bala@e-papyrus.com

Harri Oinas-Kukkonen

Department of Information Processing Science, University of Oulu, Linnanmaa, FIN-90570 Oulu, Finland. email: hok@rieska.oulu.fi

World Wide Web authors must cope in a hypermedia environment analogous to second-generation computing languages, building and managing most hypermedia links using simple anchors and single-step navigation. Following this analogy, sophisticated application environments on the World Wide Web will require third- and fourth-generation hypermedia features. Implementing third- and fourth-generation hypermedia involves designing both high-level hypermedia features and the high-level authoring environments system developers build for authors to specify them. We present a set of high-level hypermedia features including typed nodes and links, link attributes, structure-based query, transclusions, warm and hot links, private and public links, hypermedia access permissions, computed personalized links, external link databases, link update mechanisms, overviews, trails, guided tours, backtracking and history-based navigation. We ground our discussion in the hypermedia research literature, and illustrate each feature both from existing implementations and a running scenario. We also give some direction for implementing these on the World Wide Web and in other information systems.

1. Introduction

The World Wide Web's (Berners-Lee, Cailiau, Luotonen, Nielsen & Secret, 1994) brilliant success derives in part from global scale and simple access to information. Yet, web authors must build and manage most hypermedia links using simple anchors and

1071-5819/97/070031 + 35$25.00/0/hc970130

single-step navigation. In some sense, they must cope in a hypermedia environment analogous to second-generation computing languages (i.e. assembler language) in that they only have relatively low-level functionalities available. Following this analogy, we believe web authors and readers alike would benefit from the equivalent of third- and fourth-generation hypermedia features and authoring environments. Third-generation languages, such as C and Pascal, provide high-level programming concepts and data structures that enable programmers to declare procedural steps without having to worry about low-level implementation details. Fourth-generation packages such as spreadsheets, word processors and report writers let application designers (often end-users) concentrate on *what* they want to achieve instead of the process of *how* the underlying program actually produces it. Similar levels of abstraction and interfaces must be developed for the web to realize its full potential. The World Wide Web Consortium (W3C) has catalyzed the process of standardizing the web by forging a global partnership between the Internet industry and academics. The W3C wants to ensure interoperability by influencing protocol design and enforcing the quality and integrity of the web, thereby giving momentum to the development of powerful products. Our focus, on the other hand, concerns the functionality that developers should incorporate in these products [and must construct using standard web protocols in order for all web environments to have access to this functionality (Bieber & Vitali, 1997)]. In this paper, we give a vision of third- and fourth-generation hypermedia, and some direction for developers to implement them on the web—and indeed in other information systems.

We envision sophisticated application environments on the web which seamlessly integrate fourth-generation hypermedia tools with the application's other functionalities. These could include environments on the web that would support group decision making and collaborative work (Nunamaker, Dennis, Valacich, Vogel & George, 1991; Streitz *et al*., 1992; Hiltz & Turoff, 1993; Jessup & Valacich, 1993; Shackelford, Smith & Smith, 1993), authoring and guiding users through multitask processes (Scacchi, 1989; Noll & Scacchi, 1996; Tanik *et al*., 1996), educational and distance learning environments (Hiltz, 1993; Leidner & Jarvenpaa, 1995; Benyon, Stone & Woodroffe, 1997, this issue; Rana & Bieber, 1997), personal and group organizing tools (Marshall & Shipman, 1995), and enterprise coordination (Engelbart, 1990). While each of these environments has its own core set of functionality, the same types of hypermedia features, such as annotation, overviews, paths and guided tours, would serve users in all of them.

While many people think of hypermedia in terms of the World Wide Web, hypermedia research has been ongoing since the early 1960s when Engelbart developed NLS, a multi-user, distributed hypertext system (Engelbart & English, 1968). Since then, the hypermedia research community has been developing a wealth of features, systems, guidelines, frameworks and theory focusing on structuring, presenting and accessing interrelated information. We encourage builders of web technology to incorporate high-level hypermedia features, so that everyday authors and readers can utilize them. Similarly, we encourage developers of non-web-based systems to consider augmenting their applications with these features. We present several high-level hypermedia constructs which the web could provide now. We describe each, give examples from research or commercial systems (on and off the web), and note implementation considerations for each. We describe features in terms of a third- and fourth-generation authoring environment and illustrate them within a running scenario. In Section 2 we describe the

concept of hypermedia. In Section 3 we introduce our scenario and present our set of hypermedia functionalities. Section 4 concludes with some thoughts on how the hypermedia field can learn from the web community, just as the web community can learn from the hypermedia field. Our hope is that this paper will launch a major thrust to research and develop both high-level hypermedia features and the high-level authoring environments necessary for application developers, as well as everyday authors and readers, to use them. Furthermore, we hope this effort inspires researchers in other fields to put forth their own visions of sophisticated application environments, high-level features and corresponding high-level authoring environments for the World Wide Web.

A word on our terminology, which comes from the hypermedia research literature. First, hypermedia researchers view the terms *hypertext* and *hypermedia* as synonymous and use them interchangeably. Hypermedia nominally applies hypertext concepts to multiple media. Second, we separate users into two categories, *readers* and *authors*, in order to highlight how each interacts differently with a hypermedia system. We distinguish authors from the *system developers* who develop browsing software and other hypermedia environments. Readers *traverse* links during the act of browsing. Third, we distinguish the underlying *link anchor* from the manifested *link marker* displayed on the screen. Link markers visibly indicate a link's presence. Link anchors contain parameters and other internal information, which users do not see. Finally, *nodes* represent the documents or primary content containers (Halasz & Schwartz, 1994) in a hypermedia system.

2. Hypermedia

Hypermedia is a concept that encourages authors to structure information as an associative network of nodes and interrelating links. This frees authors from the linear, sequential structure that dominates most printed documents. Presenting information as an associative network enables readers to access information in the order most appropriate to their purposes, freeing them from obeying the linear ordering implied within printed documents. Many hypermedia implementations also allow readers to become authors (temporarily, at least) by adding comments (annotations) and additional links among what they read (Conklin, 1987). In all these ways hypermedia promotes options and choice.

In a larger sense, hypermedia increases comprehension (Thüring, Hannemann & Haake, 1995). Through the process of structuring information as an associative network, authors often come to understand that information better. Comprehension also increases through the enriched context that comes from sophisticated navigation support and supplemental relationships. For example, hypermedia encourages authors to provide multiple relationships around a piece of information, which readers can access directly. Thus, for readers, freedom of access within an associative structure enhanced with contextual support provides a rich environment for understanding the information they find.

Conklin (1987) identifies two major dangers of free-formed hypermedia access within an associative network: *disorientation* and *cognitive overhead*. Conklin defines disorientation as "the tendency to lose one's sense of location and direction in a nonlinear document", using the expression "lost in space" to describe it. He defines cognitive

overhead as "the additional effort and concentration necessary to maintain several tasks or trails at one time". Cognitive overhead refers to the reader's ability to follow links related indirectly to the current reading task (on a purposeful tangent or detour, or by accident), as well as the need to follow several interconnected paths to visit as much of the associative network as necessary. Cognitive theory reminds us of the overhead that comes from needing to choose among multiple links, especially for novices not familiar enough with a domain to decide among these easily (Wurman, 1989).

While many hypermedia researchers believe that disorientation and cognitive overhead are problems inherent in hypermedia, proper implementations of advanced hypermedia features will alleviate these problems, as well as provide readers with a rich information environment. In a recent survey of World Wide Web usage (Pitkow & Kehoe, 1995), only 6.54% of the 23 000 respondents reported problems with "getting lost in hypertext". While this survey was not a balanced, randomly selected sample of web users,† it does suggest that basic web browser features such as bookmarks and hot lists, backtracking and history lists and the forward, backward and home buttons provide an adequate set of navigational functions for a large group of experienced web users. Indeed, the simple backtracking function afforded by Mosaic's *back* button was shown by Catledge and Pitkow (1995) to be one of the most frequently used navigation actions. Tauscher and Greenberg (1997, this issue) actually conclude that "the only history mechanism used extensively is the Back button".

It appears that experienced readers learn to get by with the available functionality. However, the use of web technology is in part determined by its capabilities. Readers learn to make do with the available tools instead of demanding better tools, perhaps because they are not aware of how better tools might help them. Also, web browsers deal traditionally with conceptually quite unstructured information, such as prose text, images and videos. Conceptually structured data can only be represented by flattening it into (usually) text, such as the way CGI scripts retrieve relational data from a database and present it in flat text format. Web browsers have no inherent way of presenting the structure and interrelationships of data of any sort. For example,there is no way to visualize even the simple interrelationships of web documents, such as "Where can I go from here?"‡ or "Which documents point to this document?" The reader has no idea of the position of a given document within the corpora unless an author explicitly embeds such details. Yet such information is very important, as indicated by the predilection of web page authors to provide tables of contents, explicitly stating document interrelationships.

In summary, web users may not *need* features more complex than those listed above, but this paper aims to illustrate that their productivity could be improved greatly by incorporating a range of third- and fourth-generation hypermedia features to enhance web applications. Thus, while we do discuss many of the advanced hypermedia features in terms of how they reduce cognitive overhead and disorientation, we motivate each fully by showing how it enhances people's web usage.

† The authors of this survey note that the respondents did not necessarily represent a true cross-section of web users, due to the targeted circulation of the survey questionnaire and its voluntary nature.

‡ Tauscher and Greenberg (1997, this issue) note that this feature has now been added to some web browsers.

Web applications are not the only target of this paper. Indeed, even users of non-web applications could benefit from hypermedia features. Despite the rush to place information on the web, many legacy systems and other applications will keep their own interfaces, and will not be moved onto the Internet or an intranet any time soon. Yet their users still would benefit from hypermedia functionality (Davis, Knight & Hall, 1994; Bieber & Isakowitz, 1995) although they are reluctant to abandon their current non-hypermedia oriented systems in favor of hypermedia features. Thus, the myriad of today's personal, scientific and business applications, which were not designed specifically as hypermedia-oriented and do not appear on the web, should be augmented with hypermedia features. The *hypermedia functionality* approach focuses on incorporating hypermedia features into software systems so as to provide their users with an associative way of accessing, analysing and organizing information (Ashman, Balasubramanian, Bieber & Oinas-Kukkonen, 1996, 1997; Oinas-Kukkonen, 1997*a*).

In summary, the benefits of adding hypermedia functionality to information system applications are that hypermedia provides contextual, navigational *access* for viewing information and that it represents knowledge in a form relatively close to the cognitive organizational structures that people use. Thus hypermedia supports *understanding*.

3. High-level hypermedia features

Consider the following scenario:† An insurance company plans to market and sell all insurance-related products (auto, life, homeowners, etc.) to the general public through the web. Existing clients will be able to access their accounts on-line. The system also will deliver product-related information to sales agents. The amount of information to be delivered to these three audiences varies based on access privileges.

For example, information to the general public may include descriptions of products, their suitability for people in various circumstances, real-life anecdotes from clients and premiums for different kinds of coverage. Existing clients receive customized promotional information (e.g. clients who already own auto-insurance policies will find out that they are eligible for discounts on homeowner policies). Clients also have access to account statements. On the other hand, agents will have access to all the information available to the public and the clients, as well as details about commissions and incentives for selling different products; instructions, guidance and tips on selling policies and targeting clients (and potential clients), and internal policies and procedures.

Information can take the form of on-line brochures, instruction kits, application forms and short synopses and will include interrelationships. Assuming that all this information is produced by different marketing departments for different products, three different flavors for the three different audiences (clients, sales agents and the general public) have to be published for each product. Also, all product marketing information has to be approved by appropriate legal authorities within the company. In addition, the system will provide views. Agents will be able to see exactly what a particular client would see and what the general public would see. Versions of the material presented on-line have to be preserved and remain accessible for regulatory reasons.

† This scenario is discussed in more detail in Balasubramanian, Bashian and Porcher (1997).

TABLE 1
The set of hypermedia functionalities we discuss

Feature	Section
Typed nodes and links	3.1
Link attributes and structure-based query	3.2
Transclusions, warm links and hot links	3.3
Annotation and public vs. private links	3.4
Computed personalized links	3.5
External link databases and link update mechanisms	3.6
Global and local overviews	3.7
Trails and guided tours	3.8
Backtracking and history-based navigation	3.9
Other features	3.10

Interesting questions from a hypermedia functionality viewpoint include the following.

- In what ways can the insurance company effectively leverage the on-line medium by structuring the information base as an associative network with a browsing style of navigation?
- How do we provide navigation and orientation within this huge body of information?
- How do we enable a group of product marketing authors (probably new to the web and not aware of HTML, etc.) to collaborate effectively, create product-related information and establish useful interrelationships?
- How can product information be authored just once and be delivered in different views to multiple audiences?
- How do we update this information?
- How do we provide secure access to privileged information?
- How do we retain and give appropriate access to various versions of this information?
- How do we ensure link integrity between various pieces of information both in updating information content and in generating customized displays?
- In general, how do we manage this large body of information?

In this section, we address some of these questions in terms of high-level hypermedia functionality that could enhance web applications and other information systems. We ground our discussion in the hypermedia research literature, and illustrate each feature both from existing implementations and our insurance scenario. We present here only a subset of the possible hypermedia functionality which we summarize in Table 1. We briefly cover other hypermedia features in Section 3.10.

Third- and fourth-generation hypermedia encompasses both high-level hypermedia features and the high-level authoring environments system developers build for authors to specify them. Some of these features deal with node and link structure while others concern navigation tools. Many have both structural and navigation aspects. We also differentiate how authors might utilize several features in a third-generation authoring tool vs. in a fourth-generation authoring environment.

3.1. TYPED LINKS AND NODES

Semantically typed nodes and links help authors organize information more effectively and lend context for readers. Link types such as "explanation", "further details", "contrasting argument", etc., convey the relationship between the link's destination and the current node. Benyon *et al.* (1997, this issue), e.g. require "glossary", "annotation", "structural" and "associative" link types for their multimedia courseware materials. Similarly, node types categorize a node's contents. Surrounding information aside, the vast majority of web authors rely on the content of a link marker or the destination address to convey 100% of a link's purpose and destination. Yet we experience many web users complaining that sometimes they do not really know where a link will take them. This adds to cognitive overhead for the reader. We view this, in part, as both a user-interface design question and a hypermedia design question.

The gIBIS system (graphical issue-based information system) illustrates node and link typing well (Conklin & Begeman, 1989). gIBIS supports argumentation dialogues among a team of software designers, including transcriptions of design decisions and design rationale. gIBIS' conceptual model focuses on the articulation of the key *issues* in the design process. Each issue can have many *positions*, where a position is a statement or assertion that resolves it. Each position, in turn, may have one or more *arguments* that either support it or object to it. The corresponding hypertext model consists of three node types: "issues", "positions" and "arguments". gIBIS' eight link types represent the interrelations among the nodes.† Table 2 lists the link types and which node types they interrelate. Together the types create and maintain a semantics for the hypertext network designed specifically to support its domain of argumentation dialogues. QuestMap is a commercialized version of the gIBIS prototype supporting on-line planning and decision making (Group Decision Support Systems, Inc., 1996). QuestMap's interface features a semantic map linking questions, ideas and information about a project. It provides an electronic-shared "whiteboard" which teams can use during or between meetings.

In our insurance example, semantic types could distinguish a "request for policy quote form" from a "policy application form". Each node would appear as a different kind of input template. Semantic types could differentiate different types of documents which come through the regular mail, which are imaged and linked to the appropriate portions of the system. These include requests for new policies, claims, accident descriptions, police reports, complaints against agents, complaints against the insurance company, client testimonials, change of address requests and so on. Semantic types also can express a weight, context or urgency, such as "legal opinion" vs. "agent communication".

Advanced web authoring environments could take advantage of semantic types to declare such templates, i.e. skeleton pieces of applications which authors could declare and instantiate (Catlin, Garrett & Launhardt, 1991; Kaindl & Snaprud, 1991; Malcolm, Poltrock & Schuler, 1991). Templates can combine nodes and links that go together. For example, when a user wishes to comment upon an existing node in gIBIS or QuestMap discussions, the system presents the set of legal link types from that node and the set of legal node types for each link type. Choosing one displays an empty template form to fill in, tailored to that node type. Bieber & Vitali (1997) briefly describe a Delphi methodology

† gIBIS has a ninth link type, "unspecified", which is not discussed by the authors in any of their papers.

TABLE 2
gIBIS's enforced node and link structure

Node type (source)	Permissible link type(s)	Node type (destination)
issue	generalizes, specializes, replaces, questions, is_suggested_by	issue
issue	is_suggested_by, questions	position
issue	is_suggested_by, questions	argument
position	responds_to	issue
argument	objects_to, supports	position

(Turoff & Hiltz, 1995) group consensus system in which authors use hypermedia templates to construct the survey questions.

Most hypermedia systems that provide semantic types, such as SEPIA (Streitz *et al.*, 1992; Thüring *et al.*, 1995), LIRMM's MacWeb† (Nanard & Nanard, 1995) and gIBIS, display the types as labels within an overview diagram. Link labels appear next to the arrows that represent links, while node labels may appear either as the text of the node or next to the node. In NoteCards, authors optionally can represent links as boxed labels positioned anywhere within the node's white space (Halasz, 1988). Few systems embed labels in the text itself next to the link marker. Although this should be possible—distinguished in some manner, such as delimited by ⟨angle brackets⟩—it could clutter up the node's existing contents. Alternatively, link labels could pop up when the user moves the mouse cursor over them (Vitali, Chiu & Bieber, 1997). A hypermedia version of papers from the August 1995 issue of the *Communications of the ACM* (Bieber & Isakowitz, 1995) designed for the current Netscape viewer employs the latter approach (SIGLINK, 1994; Bieber, 1997). Articles appear in the main window frame. Moving the cursor over a link marker displays the link's semantic label as well as *structural information* about the destination (article title, author, section, type, etc.) in a second frame. Another frame maintains semantic and structural information about the link traversed to arrive at the current node. Thüring *et al.* (1995) list providing semantic link labels as their first principle of hypermedia design.

Some systems display labels to help the reader distinguish among multiple links from the same link anchor. While Intermedia (Yankelovich, Haan, Meyrowitz & Drucker, 1988), ACM's Hypertext on Hypertext (Yankelovich, 1988) and Microcosm (Davis, Hall, Heath, Hill & Wilkins, 1992) only display labels in pop-up menus (preview lists) when readers select anchors with multiple links, Max (Bieber & Kimbrough, 1992) always displays the link type, even for single-destination links, so the reader can choose whether to follow it. Microcosm also has this option. Vitali *et al.* (1997) propose a simple method called *displets* for authors to include pop-up boxes with link labels for multiple links (as well as many other features that could be specified from within HTML).

† MacWeb is a knowledge-based hypertext system under development at LIRMM since 1989. It has no relationship to the World Wide Web client developed later with the same name.

A third-generation type of authoring tool would provide authors with a data structure containing a specific parameter for semantic types, and the tool would display node and link labels automatically in overview diagrams and in other content displays. A fourth-generation authoring environment would add a magnitude of functionality. Authors would be able to choose from a set of established node and link types, and would have a high-level interface for creating new ones (or new sets for a specific group of other authors). The system would associate semantic types with appropriate *browsing semantics* (Garzotto, Mainetti & Paolini, 1996; Schwabe, Rossi & Barbosa, 1996), i.e. the behavior associated with objects of that semantic type. For example, gIBIS would enforce its semantic structure. Following Table 2, authors only would be allowed to attach certain types of nodes to a particular link, and specific types of links between two particular nodes. Indeed, authors could just choose from a menu item "I wish to challenge this position" and the system would automatically create the correct link and open up the appropriate node type for the author's comment. In the insurance example, selecting a generic "client-view", "public view" or "agent view" link (each automatically attached to all documents) would generate the appropriate view, assuming the user has the correct access permissions (see Section 3.4). A client requesting to file a claim would automatically receive the correct "claim form" node template. Once complete, the form could follow an established workflow path, mailing itself to the appropriate claims officer for approval. If the form then must go to others, it would automatically route itself accordingly. The fourth-generation environment would allow authors to specify these behaviors based on node and link types at a very high level.

Textnet (Trigg & Weiser, 1986), SEPIA (Streitz *et al*., 1992), MUCH (Wang & Rada, 1995) and Debate Browser (Oinas-Kukkonen, 1996) enforce different sets of link types. Other systems, such as LIRMM's MacWeb, NoteCards and Linking Ability (Oinas-Kukkonen, 1997*b*) permit arbitrary link labeling. MacWeb, as a knowledge-based hypertext development environment, allows authors to type links, nodes, groups of nodes, multimedia items within nodes and link anchors. Not only do these give authors a high degree of semantic specification, but authors can create scripts based on any of these types. Arbitrary labeling gives authors more flexibility to express exact meaning, but carries the danger of inconsistent type names (Rada, 1990). Wang and Rada (1995) also point out that authors often do not take the time to incorporate link types, even when appropriate, so system designers providing semantic typing should make them as easy to specify as possible.

Even when the reader's hypermedia environment does not display node and link types, Nanard and Nanard (1995) encourage system designers to facilitate typing as far as possible along the design process to assist authors in structuring and organizing their work. Bieber similarly encourages application developers to determine the semantic relationships within an application as part of the systems analysis and design process, even if users will not gain direct access to these relationships through links in the final application (Bieber, 1996).

The hypermedia research literature has produced other link taxonomies designed purely for the benefit of researchers and system developers, which systems will never display to readers (DeRose, 1989; Rao & Turoff, 1990; Parunak, 1991*a*; Bieber & Vitali, 1997). These include categories such as extensional, inclusive, intensional, implicit and isomorphic (DeRose, 1989). System developers can use these taxonomies in determining

appropriate navigation strategies and ways of conveying context for different categories of links underlying fourth-generation environments. Rao and Turoff also include a theoretical node taxonomy for the same purpose. In fact, Rao and Turoff have suggested a general semantic framework for hypermedia functionality based on Guilford's Structure of Intellect model. This framework classifies nodes into six different semantic types and links into 12 different types. Nodes represent cognitions or ideas. Links represent the relationships between ideas. Links are further classified into convergent links and divergent links. Convergent links help in focusing or narrowing the pattern of relationships between ideas whereas divergent links expand or broaden them.

Actually, the web has always had a mechanism for specifying link types, and has even a set of types ##(*rel-rev*), which slowly is becoming an accepted standard (Maloney, 1995). Few browsers, however, take these into account. The CLASS, REL and REV attributes of HTML's A and LINK tags allow authors to specify a complex typology of links, while the TITLE attribute helps attach a text label to the anchor. The REL attribute specifies the relation with the link's destination. REL values include support for sequences (NEXT, PREVIOUS), hierarchy (CHILD, PARENT, TOP), navigation (TOC, INDEX), home-defined links (HOME, BACK) and common types of relations (CITATION, DEFINITION, BIBLIOENTRY, FOOTNOTE, AUTHOR, COPYRIGHT, DISCLAIMER, etc.). The REV attribute contains, where appropriate, the reverse relation (e.g. an anchor with REL = NEXT will have REV = PREVIOUS). The CLASS attribute contains free text further describing the link's type (so that it becomes possible to define several links with REL = NEXT, e.g. each with a different CLASS—NOVICE, INTERMEDIATE, EXPERT).

Unfortunately, discussions within the HTML standards working group cover the acceptable syntax and values of these keywords, but make no mention of implementation or use, leaving these to web editor and browser developers to figure out. Web browsers, e.g. could display semantic link types in several different ways, including in pop-up windows or in separate frames as with the aforementioned *Communications of the ACM* issue. Web-based editors could make it easy for authors to assign a semantic type to their links, both through pop-up lists of available links and by devoting a special input field in the link dialog allowing for new type specifications. While authors must include *semantic* types as a parameter, systems could infer most *structural* information from the HTML/SGML (International Standards Organization, 1986) layout of the link's source and destination documents. The proposed displets by Vitali et al. provide support for exact rendering of new HTML elements (Vitali *et al.*, 1997). The displets (special Java classes) are called to perform the display of newly defined HTML tags, thus allowing fine control over appearance of the document elements, while maintaining the descriptive markup of HTML.

3.2. LINK ATTRIBUTES AND STRUCTURE-BASED QUERY

In addition to types, other semantics can be attached to links, such as labels, names, keywords or timestamps. A single link type defining a conceptual relationship between nodes can have multiple labels representing the type tailored for different contexts (Rao & Turoff, 1990). The keywords attached to a link can include its type and labels. All

users—including readers—may be allowed to define keywords on links. Any number of keywords can be attached to a link.

A fourth-generation authoring environment would provide link customization features. The Linking Ability system enables team managers to define link sub-types and keywords for each team. Authors then choose link sub-types from these (Oinas-Kukkonen, 1997*b*). In our insurance scenario, an author might want to create an annotation link of sub-type "legal opinion" with keyword "highly important". Admittedly, defining attributes takes effort, but the payback comes when readers receive structural information in vast information collections, distributed environments or application domains, in which they can see and exploit dependencies among pieces of information.

While many hypermedia systems provide a text string search facility for node content, they lack a *structure-based query* based on node and link attributes (Halasz, 1988; Lucarella, 1990; Lee, Yoo, Yoon & Berra, 1996; Maurer, 1996). Using the insurance company example, a team manager may want to find the following.

- All complaint letters (letters with a "complaint" semantic type or keyword) about a particular insurance agent.
- All "legal opinion" links created about his/her product line since 1 January.
- All links with the keyword "highly important".
- All links created by a particular insurance agent.

Executing the query results in a hit list consisting of the links that satisfy the query or an overview diagram of the system with just these links highlighted. Readers can inspect the dependencies between documents (Bieber & Vitali, 1997). Implementing structure-based query will be facilitated greatly by maintaining external link databases (see Section 3.6).

HyTime (Newcomb, Kipp & Newcomb, 1991; DeRose & Durand, 1994)—the hypermedia extension to SGML—contains provisions for using structure-based queries to determine link endpoints. The first version of the HyTime standard contains a query language, called HyQ, which can be used to identify document elements satisfying given structural properties (such as "the third paragraph of all annotations created after 3/31/96"). Although a few partial implementations of HyTime engines exist, the potential of queries as part of location specification has barely begun to be realized.

Attributed links and structure-based query becomes especially useful in complex, knowledge-intensive domains that emphasize teamwork, such as collaborative software design and inspection (Tervonen & Oinas-Kukkonen, 1996). As Tervonen, Kerola and Oinas-Kukkonen (1997) describe, collaborative hypermedia functionalities can provide a solid platform for the growth and intuitive use of an organizational memory (Stein & Zwass, 1995).

3.3. CONNECTING OCCURRENCES: TRANSCLUSIONS, WARM LINKS AND HOT LINKS

Transclusion, warm links and hot links all connect two occurrences of the same information. They differ in philosophy, functionality and implementation.

Transclusions (or inclusions) was one of the first hypermedia features to be proposed by hypertext visionaries, but thereafter ignored by hypermedia implementors. Transclusions constitute the core concept within Ted Nelson's proposed system Xanadu (Nelson, 1987, 1995). Nelson proposed transclusions as a mechanism for having the exact same

object (document content) exist in multiple places. Whereas copying and pasting creates an identical copy, transclusions act similar to pointers that connect the original copy to all places that use it. Transcluded data is *alive*, still connected to the original and automatically updated. Through transclusion, readers always have access to the original and therefore to its original *context* (through a context link). Xanadu's virtual document structure is built around transclusions: each document is a list of pointers to pieces of data, which originate in that document or are "included" from others.

In the context of our insurance scenario, transclusions create a reliable and maintainable set of similar documents (e.g. a customized document containing policy information for a customer, including the policy's payment and payout schedule). The complexity and number of contracts inevitably results in a huge amount of documents containing the same basic items, but each different in a specific way. Storing each document as a separate, complete entity would mean vast redundancy and complex update whenever a fairly common clause in many needs to be changed. Using massive transclusions could reduce or eliminate redundant storage and greatly simplify the update process. Authors would specify documents in terms of transcluded elements. For example, a document may comprise standard clauses 1 to 12, clause 13 from document Y, and a special clause 14 belonging to only this kind of document. The same thing can be accomplished by component documents and republishing all documents that use a component when that component changes. [Xanadu's implementation of transclusion inherently incorporates versioning (Vitali & Durand, 1995), which would satisfy the insurance company's need to maintain old document versions.]

Transclusions are related to but differ from composites. Whereas composites intrinsically refer to whole nodes (Halasz, 1988), transclusions refer to sub-parts of a node. Thus, the problem with composites is not with composition but with decomposition: the granularity of the atoms forces the minimal granularity of a composite's elements. It is thus impossible to include in a composite a chunk of text smaller than the full granularity of the text's unit. This affects the types of composites possible. For instance, authors could build the policy documents using composites; storing each clause as an atomic object and creating standard "sets" or collections of clauses would allow authors to compose modular documents very easily. But in the case of two documents differing only in a part smaller than a clause (e.g. the policy term), authors would be forced to create two distinct clauses identical in everything except the term, thereby losing the notion (and operational benefit) that these clauses essentially are identical.

Warm and hot links are relationships that create a channel between the two endpoints, through which data flows from one document to the other (Meyrowitz, 1989). Warm and hot links are not pointers, but actual copies of the data, which can update automatically. With *warm links*, users explicitly ask to update the node content. With *hot links*, content is verified and updated automatically as soon as the data are displayed. Transclusions and warm linking differ basically only on implementation details. Transclusions are pointers to data. With warm and hot links, the host document contains both a copy of the data and a reference to the original source. Simple references avoid duplications and redundancies, while copies allow improve realiability and availability—the system could still display a (possibly outdated) copy even if the original document were deleted or inaccessible.

Curiously, while few existing hypermedia systems have ever implemented hot and warm links (Catlin, Bush & Yankelovich, 1989), implementations have come from an important class of commercial systems: operating systems. Publish/subscribe in the Macintosh operating system and dynamic data exchange (DDE) in Microsoft Windows allow applications to create live channels for displaying portions of data from one application to another. The difference between transclusions (or warm/hot links) and composites can be shown clearly by noting composites' parallel to UNIX *symbolic links*, MacOS *aliases* and Windows'95 *short cuts*. Just like composites, symbolic links allow containers (i.e. directories) to share the same components without duplication and just like composites, only whole files can be shared among containers. DDE, on the other hand, allows arbitrarily sized chunks of data to be shared among documents, as transclusion would dictate.

A form of transclusion exists in the web environment. Through *server includes* authors can create documents, the content of which is decided dynamically, just before the document is sent over to the client. Within the HTML code, authors can include special commands that the server parses, substituting the resulting values every time the client requests a document. Commands include invoking CGI applications, inclusion of whole files and substituting variables. Although potentially quite powerful, currently authors use this mechanism primarily for simple variable substitutions, such as the number of visitors to a web site, or placing standard headers and footers automatically on all documents belonging to a site.

3.4. ANNOTATIONS AND PUBLIC VERSUS PRIVATE LINKS

Hypermedia researchers have always considered private annotations (comments) a basic right for hypermedia readers as well as a basic tool for collaboration and exchange of ideas. Almost every major hypermedia system provides support for annotations: KMS (Akscyn, McCracken & Yoder, 1988; Yoder, Akscyn & McCracken, 1989), Intermedia (Catlin *et al.*, 1989), Aquanet (Marshall, Halasz, Rogers & Janssen, 1991), the Virtual Notebook System (Burger, Meyer, Jung & Long, 1991), StorySpace (Joyce, 1991), etc. While web committees have discussed annotations (La Liberte, 1994; Gramlich, 1994), few browsers support annotation as it is difficult to embed annotation links with HTML documents belonging to others.

Mosaic 2.4 (National Center for Supercomputing Applications, 1995) provides an annotation facility. Selecting the "annotation" menu item brings up an edit window into which the reader enters the annotation text. Committing the annotation saves its content to a file in the reader's personal workspace, attached by default to the currently displayed document. Henceforth, every time the reader displays this document, a link to the annotation will appear at the bottom of the document. But unlike most hypermedia systems, readers cannot associate an annotation with the specific spot in the document upon which the reader wishes to comment. This potentially increases both disorientation and overhead. However, it would be a simple matter to alter this to operate as open hypermedia systems do, associating the annotation with a specific position without embedding it in the document (see Section 3.6).

An additional issue is whether to allow users (and, especially, *which* users) to customize the data, the way it is displayed or the available links. Creating views and only permitting

certain readers to access certain information requires customization based both on user role and access permissions. While restricting access is an anathema to hypermedia purists, real-world organizations demand such security. Our insurance company will not want to run the risk of disgruntled employees or the general public adding negative annotations on the annual report. Similarly, only employees of the legal department should be able to add links to certain documents where available information must be within exact legal guidelines. Bieber and Kacmar (1995) give additional examples for the domain of geographic information systems.

Besides standard read and write access permissions on documents, system administrators may consider analogous read and write permissions for links and annotations. For certain documents, one could allow link authoring and update while prohibiting write access to the underlying content. One also could restrict author access to certain node and link types. For example, only employees of the legal department could author a "legal opinion" link. Furthermore, one could regulate link and annotations to specific views.

Within the ABC collaborative writing environment, access permissions regulate group collaboration instead of acting philosophically to prohibit free communication (Shackelford *et al*., 1993). Turoff has developed 25 kinds of access permissions (as well as "tickets" to override them) for computer-mediated communications, many of which would apply to collaborative hypermedia applications (Turoff, 1991). In fact, access permissions may prove most useful for allowing individuals to maintain a personal set of nodes, links and annotations, while collaborating work groups would maintain a shared set of these. HyperWave uses access rights assigned to links to implement private and group networks (Maurer, 1996).

Mosaic's annotation facility supports three classifications: personal, workgroup and public annotations. Later versions support workgroup annotations through a server providing annotations to any person who has the appropriate access permissions. The Distributed Link Service implemented a similar approach (Carr, De Roure, Hall & Hill, 1995). In each case, the document server and link server can be completely different entities. This requires the link server to employ externalized link databases (see Section 3.7).

With personal and workgroup annotations, annotation authors implicitly are trusted once they log onto the supporting network; hence, there is little need for annotations to be vetted for suitability. A public annotation facility, however, may or may not permit annotations to be added by an unvetted public. Thus, these systems distinguish read-only public annotations from unrestricted public annotations. For example, a site may publish a database of annotations or links which cannot be augmented by the public (as the Distributed Link Service does). Alternatively, an annotation/link server may permit readers from anywhere to add publicly readable links and annotations to any document. These two classifications, respectively, are directly analogous to the concepts of moderated and unmoderated newsgroups. For this very reason, implementors of public link and annotation databases may find it profitable to consider the techniques of network bulletin board management (such as the use of "kill" files to filter out irrelevant or obnoxious annotations). For instance, our insurance company may publish a page calling for clients to make comments about the service provided by the company, yet they may be understandably reluctant to make public any negative

comments. Thus, they could implement a form of moderated public annotations for this purpose.

Schloss (1996) presents an annotation mechanism on the World Wide Web for augmenting electronic commerce applications with advisory systems. His advisory service enables third parties to provide supplementary information to support, augment or critique the contents of web pages. Organizations will be able to offer rating services (e.g. by special interest groups or along the lines of *Consumer Reports*), active advice (e.g. discounts for preferred customers) and related information (e.g. relevant magazine articles). We anticipate this functionality to have significant social impact. Schloss' advisory service, as well as external link databases (see Section 3.6), render many security issues irrelevant, but organizations still will need content and annotation control on intranets behind company firewalls.

3.5. COMPUTED, PERSONALIZED LINKS

An important and powerful feature of the fourth-generation hypermedia systems is the automated creation of private links from computations. This necessarily means that the reader must be able to specify their own computations in order to create the private links, i.e. readers must be able to tailor their own link computation specifications. This allows readers to create entire collections of links which are specialized to their needs.

For example, a sales agent in the insurance company may wish to create a personalized link computation specification which will take the name of any customer and link it to every insurance policy ever held by that customer. Rather than have to create this set of links by hand every time a policy is sold, the sales agent sets up a sort of "standing order" with the hypermedia system, so that it will automatically create these links whenever a new policy is taken out.

Tailoring of link computation specifications is not well supported in general. There are few hypermedia systems which permit users to create their own link computation specifications, and not many more which permit administrators to create link computation specifications for corporate use. Where this has been enabled, the specifications for the computation of links from one link type are expressed in some formal language. One implementation of tailorable link types uses a relational database to store these specifications (Verbyla & Ashman, 1994). The web's link computation specifications (i.e. CGI scripts and mobile code applets) are usually stored in directories on the host machine operating system and hence are not immediately tailorable. However, the advent of client-supplied code in the form of servlets (Sun, 1996) will enable unlimited link-type tailorability. Alternatively, computation specifications can be attached to classes of objects, and while the reader may not necessarily be able to alter the class (i.e. the domain to which the link type applies), it is possible to edit specifications for computing the destinations of links (Monnard & Pasquier-Boltuck, 1992; Rizk & Sauter, 1992).

The key issue in computing private links is that the computation specifications do not form part of the software of a hypermedia system, hence they can be manipulated without the need to recompile (and hence alter) the software. This allows both readers and system administrators to create and maintain link computation specifications most relevant to their needs. This is a form of externalized link database, except that the stored objects are not actual links but are specifications for computing links.

3.6. EXTERNAL LINK DATABASES AND LINK UPDATE MECHANISMS

Embedding link endpoints, as in HTML, presupposes that the link creator has write access to the information being linked. External links, which are stored in external *linkbases* separate from the document content are used where write access is not available or when user tailorability of links is supported. For example, the field of *open hypertext systems* (Wiil, 1997; Østerbye & Wiil, 1996) uses external links for linking to data owned by legacy applications and other non-hypermedia applications without disturbing the original format of that data (Davis, 1995*a*; Kacmar, 1996). We also see in this paper that maintaining external linkbases supports link and view customization, guided tours, and trails.

A typical situation would be where users want to apply their own link specifications to data from a remote site or from read-only media, or where documents being linked must remain inviolate. For example, our insurance company may wish to link client correspondence to the client file. For legal reasons, the correspondence must not be altered in any way; hence, links coming from the correspondences must be maintained externally. Another case where documents must remain inviolate is for accountability purposes. For example, links to previous versions of a product brochure must be maintained for historical, legal or temporal context. The advertising department may wish to link the advertising pages of other insurance companies to its own advertising pages when it believes the comparison in its favor. Since our insurance company is most unlikely to get permission to create such links within their competitors' pages, they must use an externalized link database.

Microcosm (Davis *et al*., 1992) implements every linkbase to be external and independent of all the documents it connects. In addition to any communal linkbases, all users have their own linkbase which they can carry with them on a floppy disk. Readers have access to all links in all available linkbases. This shows the feasibility of private linkbases containing links connecting public documents. External link databases are being tested on a larger scale in the Distributed Link Service (Carr *et al*., 1995), in which databases of precomputed links are published along with documents. A reader can connect to a server to retrieve documents, but (with the appropriate browser or preprocessing engine) can also connect to a server to retrieve links appropriate to a document. The HyperWave distributed hypertext system (Maurer, 1996) employs external link databases in order to provide bidirectional point-to-point links within multiple media with rich typing, individual access permissions and guaranteed consistency. Whenever documents are removed, every link pointing to it is removed from view (i.e. it is not deleted, but it becomes hidden to readers). The Gentler system (Thimbleby, 1997, this issue) uses an external link database for consistency maintenance, embedding links in the HTML code only when creating the final web page for display.

To support links, endpoints of which occur at specific named points within documents, links must include not only document names and network nodes, but also internal locations. The addressing schemes of Xanadu and Microcosm both refer to internal locations in documents using a document name/byte offset form [much like Xanadu's tumbler addressing scheme (Nelson, 1988)]. Other systems dynamically compute link locations with e.g. string-matching algorithms. These links are either reconciled to the document by a special-purpose browser, or the HTML is amended by some

preprocessing agent before it is tendered to a plain browser. While "published" link databases are intended for widespread consumption, there is no reason why a user's private link database should not be used in the same way.

Not embedding links faces the problem that information update can desynchronize the link's reference in a linkbase and its new location in its document. Before using it, one needs to check whether the link is still consistent. This means checking link *validity*, i.e. whether the nodes still exist, whether they are still reachable and whether the stored references still point to the exact location within the nodes. It also means checking link *relevance*, i.e. whether the link still has a reason to exist or, more exactly, whether the changes in the documents have made the link useless, inappropriate or wrong. These problems arise when there is loose control between the link and the connected nodes, i.e. when a change in one of them does not necessarily imply a modification in the others.

Our insurance company can end up in difficulties when URLs become invalid. It may have taken out a large paper advertising campaign, distributing leaflets which all have the company's home page URL. But if that URL becomes invalid for whatever reason (such as a change of Internet service provider), the company runs the risk of alienating a number of existing and potential customers.

Our insurance company also can run into problems when link relevance is compromised. For example, a competitor may decide to undercut our insurance company in price and suddenly change the contents of their advertising pages without warning or notification. The link which our company uses to show that its price is better will henceforth demonstrate exactly the opposite! Validity of links, on the other hand, is at stake when the competitor rearranges the content of the document so that the link points to an irrelevant part of the document, leaving readers wondering about the link's purpose.

The World Wide Web is not immune from the problem of updating links. HTML uses internal links, i.e. links that have endpoints explicitly stored along with the text, with ⟨A HREF⟩ tags as sources and ⟨A NAME⟩ tags as destinations, with reference information about the destination stored in the first tag. This reference is composed of a URL (a locator of the destination) and the optional name of the end-anchor. The URL, anchor names and document contents all are controlled by the destination document's author, whereas the ⟨A HREF⟩ tag is controlled by the source document's author. Since there is no tight connection between the two documents, users find links failing after any change in name of the anchor, the destination file or of the network node, since embedded links (in this case URLs) are not updated automatically.

Alternatively, suppose the document's location remains the same, but the contents change. While byte-offsets make it possible for users to create links on documents they do not own, byte offsets fail after even minor modifications to the content (for instance, offsets are incorrect after any insertion or deletion preceding them). While missing the real endpoint by a few characters may seem a minor problem for navigation links, it may become a vital problem if transclusions are realized in the scale of the web. Because transclusions affect the actual content of the node containing the reference, if the reference has not been corrected after the referred to document has changed, then the content of the node, and not just the link, becomes unreliable.

Universal resource names or URNs (URN Implementors, 1996) should resolve these problems partially. URL (Universal Resource Locators) and URN are two types of URIs

(Universal Resource Identifiers), which are addresses used by web software to retrieve and access to network resources (World Wide Web Consortium, 1996). A URN provides a reliable and persistent address to a network resource. Whenever a URN address is requested, a network resolver determines the current and most appropriate actual address (most often, a URL) for the client to request. URNs most commonly comprise a directory system allowing users and servers to register documents and other resources. The directory service would then provide a persistent URN that could be reliably used in anchor tags. Whenever a modification occurs, the person responsible for the document would update its information in the directory service, so that any subsequent access to the persistent URN name would lead the user to the updated location. For the system to work, document owners need to register them with a nominated document registry and update the registration whenever the location changes. Link creator must remember to use URNs instead of URLs. Of course, our insurance company's competition may decide not to register a URN specifically to avoid persistent tracking, either by our company or by a negative advisory service (see Section 3.4).

The URL addressing scheme could be seen as a second-generation addressing approach, since it requires an accurate knowledge of the storage details of the document. URNs can be considered a third-generation addressing approach since the storage details are concealed from the author, although the author still needs to know the unique identifier assigned to a document.

Different philosophies (implemented in different systems) address the link update problem.

- *No change.* Information is decreed to be stable, with change and deletion not permitted.
- *Notification.* The agent that changes or deletes data issues warning notices to all applications which have precomputed links over that information. For instance, HyperWave employs a server–server protocol in order to guarantee consistency on links across server boundaries as soon as the nodes are modified (Kappe, 1995). Note that the notification scheme cannot be guaranteed to work for large-scale external link databases. Since linkbases can be activated and deactivated at will, and indeed can go off-line at any moment (e.g. the volume containing it can be dismounted), control over linkbases is practically impossible, thus making it impossible to track all links that point to or from a document and update them. Besides, in a truly worldwide information collection, the effort required to notify every linkbase of changes would be prohibitively expensive, because links could be stored in thousands of linkbases around the world. The best compromise is to use URNs so that only a limited, identifiable number of repositories need to be notified.
- *Regular updates.* Information changes at predictable intervals, i.e. information is issued in "editions". Notification of changes is now optional, as change can be predicted.
- *Detect and correct.* The application using externalized links first checks the validity of the link reference against the information, perhaps matching with an expected value. If the validity test fails, an attempt to correct the link by relocating it may be made. This strategy resembles the dynamic linking strategy below, since it involves a computation to validate the endpoint of the link. An example of its use is the

"conditional get" in the http protocol (World Wide Web Consortium, 1995), in which the date-stamp (or similar attribute) of the cached version of a document is compared to the original document, resulting in a retrieval of a new copy if the two are different (involving a connection to the document server in any case).

Vanzyl, Cesnik, Heath and Davis (1994) proposed heuristic techniques to retrieve the current position through pattern matching, based on the fact that insertions and deletions in parts of a document far away from the link endpoint usually do not affect the context around the endpoint itself. Thus, the endpoint's unchanged content can be found again when needed. While simple and powerful, the heuristic is not perfect, and fails in certain situations such as multiple occurrences of the context or link text when it cannot distinguish which instance we want.

- *Dynamic linking.* Assumes that all information is subject to change without notice, i.e. links will not be pre-computed, but rather computed as needed. Applications which provide a hypermedia interface to relational database information use this technique because of the volatile nature of the information (Verbyla *et al.*, 1994). In other cases, the link destination may be affected by each reader's current context, e.g. the current time and date will affect links to news stories.

  Many versioning methods use a dynamic linking approach, e.g. Maioli, Sola and Vitali (1993) and Nelson (1987). Being able to check what changes have been performed on a document allows versioning systems to compute current offsets based on the older ones. In fact, the Xanadu addressing schema, heavily based on versioned transclusions, purportedly (Nelson, 1987) never runs into the link update problem because original offsets are always valid regardless of how a document is modified.

Fully implementing many fourth-generation hypermedia features, such as views, private link collections and transclusions will rely on external linkbases. Developers ought not be discouraged by the link maintenance problem. Versioning is absolutely reliable as it employs a form of the "no change" approach where document versions are never deleted or altered—new versions are constantly added. Authors, however, will be able to use only versioning-aware tools when creating and modifying links. A versioning mark up language for the web has been proposed (Vitali & Durand, 1995), and its implementation in HTML editors and browsers would contribute to a natural and widespread use of reliable update techniques.

3.7. LOCAL AND GLOBAL OVERVIEWS

Overview diagrams (or maps) provide a view of the hypermedia node and link network from above. They show its structure in a graphical manner (e.g. with icons and arrows). Global overview diagrams provide an overall picture and can also provide anchors (entry points) to local overview diagrams. Local overview diagrams provide a fine-grained picture of the local neighborhood of a node. Both can serve as excellent navigational aids. Overview diagrams or maps, both at the local and global levels, improve spatial context and reduce disorientation in a hypermedia network. Local overview diagrams also help minimize cognitive overhead by showing a small, relevant part of the network.

Intermedia provides overview diagrams, called "web views", which provides both spatial context ("Where can I go from here?") and temporal context ("When and by which path did I get here?") (Utting & Yankelovich, 1989). Intermedia web views are dynamically updated, local tracking maps that display all documents linked to the current one. This provides local context as well as link previewing. Also, selecting a link marker in the full content of a node highlights the corresponding link line in the adjacent web view, giving the user an idea of the target without actually following the link. In Thoth-II, the "Spiders" directed graph browser provides a contrasting type of overview map (Collier, 1987). As the reader interacts with the structure being viewed, new graphic objects (nodes and links) are created. Activating a node expands it to show links to other nodes which subsequently fan-out to further nodes. The network expands or fans out in two-dimensional space creating "spiders" on the display. The window on which this structure was displayed could be moved around to view the parts falling outside the viewing area. Bieber and Kacmar (1995) also provide examples of overviews within a geographic information system. The HyperWave system (Maurer, 1996) provides *customizable* local maps, in which the user can specify the depth of the map, i.e. the number of steps that are displayed, starting from the selected node.

Overview diagrams can be generated either by reading the links between documents from a link database (see Section 3.7) (Andrews, 1996) or by examining the documents themselves for embedded links. However, as soon as overview diagrams for large systems become complex they might introduce navigational problems of their own (Nielsen, 1990). Nielsen suggests features which we believe an intelligent fourth-generation overview map would provide. For example, in order to reduce the propagation of links from the local diagrams to global diagrams, weights can be assigned to the links based on their relevance to the user; this will trim the edges of the graph at the global level. Similarly, readers might want to know what proportion of the material in the current sub-web applies to their goal. [The scope line in Intermedia, e.g. informs the reader about the number of documents and links in the web (Utting & Yankelovich, 1989).] The overview browser should apply user profiles to the node and link attributes, perhaps querying the user for his or her interests, and then tailor the overview appropriately and automatically.

In our life insurance company scenario, the amount of material to be presented highlights the need for overview diagrams. An agent may be interested in finding out the local and global context of a particular product. Is a product part of a category of products with similar characteristics? Is that category part of another higher-level group of products? Is the product itself related to other products from other groups? This kind of information is hard to assimilate when readers only can function down at the actual content level, but much easier when the relationship (meta)structure is shown in a graphical manner.

The World Wide Web does not have the necessary constructs to provide local and global overviews or maps for two unrelated reasons. First, while global overview maps can be provided in terms of image maps, their automatic generation would be extremely complex and troublesome, having to deal with CGI or Java applications parsing the documents for link information, and creating the image (possibly a GIF) and the image-map table for relevant users' feedback. Second, web links cannot be categorized by their roles; there is no easy way to differentiate between structural links among parts of single concept or information unit (e.g. the various details of an auto-insurance policy)

and associative links among related concepts, (e.g. relationships between two products such as auto-insurance and homeowners' insurance). Furthermore, many HTML authoring tools do not allow authors to visualize the nodes and relationships which they are creating, although tools such as Microsoft FrontPage and NaviPress allow authors to look at local mini-webs as they are being created.

3.8. TRAILS AND GUIDED TOURS

In what people now recognize as the first article on hypermedia, Bush† introduced the idea of associative *trails* (Bush, 1945) or *paths*. Trails connect a chain of links through information spaces. They provide a context for viewing and understanding a series of documents. Trails can record a path of information that the reader may wish to remember and share with others. Authors (or interested third parties) can prepare multiple "recommended" trails through an associative network focusing on different aspects or tailored to different readers (a novice, an expert, a teacher, a student, etc.). Bieber notes that analysts might use trails to document a decision analysis for management or colleagues (Bieber, 1992). Continuity and guidance distinguish trails from random links in documents. The trail should be clearly marked, so that users will know which links keep to the trail and which constitute *detours* from the trail. The trail designer could filter out links, making only the ones most relevant to the current item available as detours. *Guided tours* restrict users to the trail, prohibiting detours. Nodes viewed during the guided tour will have links off the tour dimmed or hidden. Users have to suspend or exit the tour to access these (Garzotto *et al*., 1996). While trails lower cognitive overhead by recommending the next logical link to take, guided tours reduce overhead further by removing all other choices. This notwithstanding, trails and tours could contain branches allowing the reader to pick one sub-path over another (Thüring *et al*., 1995). In addition, the system could provide an overview or map, so readers can maintain their orientation along the trail or tour.

We can imagine endless opportunities to enhance our insurance company scenario with trails and guided tours. Agents can see all correspondence from a particular client regarding a particular claim in chronological order. The company could establish a guided tour for the general public about each of its major products and then nested guided tours within this about particular product details. An agent might prepare a customized tour for a client or potential client. The training department could set up tutorials for new agents and those seeking advanced training in various topics in the form of trails. Trainees could follow the recommended path while having access to details and tangential information.

Trail implementations vary in complexity. In simpler implementations, such as with Textnet, the trail is an ordered list of nodes that the user follows. Textnet trails can be viewed in a special window as a list of nodes. Both the author and the reader can change this list at will to reflect new discoveries or themes (Trigg & Weiser, 1986). Perseus represents trails as an ordered list of icons (Mylonas & Heath, 1990). The editor allows authors to

† Vannevar Bush was Director of the Office of Scientific Research and Development during World War II for President Franklin D. Roosevelt. Brown University recently sponsored a 50-year symposium to honor his work (Brown Computer Graphics Group, 1995; Simpson, Renear, Mylonas & Van Dam, 1996).

build path entries to entire nodes or to a span of text within a node, to rearrange path entries and to annotate entries with brief comments. In Intermedia, a path was a list of documents readers visited earlier in a browsing session. The display of a path consisted of the name of the document, an icon indicating the type of event (opening or activating documents), and a timestamp indicating when the event occurred. A user's path could be saved when closing the web and restored when subsequently opening the web. Thus, a path can be used to collect all interesting documents to form a linear document that can be preserved in printed form (Utting *et al.*, 1989).

More advanced implementations permit the author to annotate the trail with fuller information about its overall purpose and execution, as well as about each of its nodes. For example, NoteCards' (Halasz, 1988) guided tours feature a graphical overview and customized tabletop screens at each stop along the tour (Trigg, 1988; Marshall & Irish, 1989). The tour's author can add the following types of metainformation in a separate area on the tabletop for the tour and for each of the *notecards* or nodes included on the tour.

- Explanation of each notecard's content.
- How to interpret the tour's screen layout (parameter settings; what each font size, division of screen space and pointer arrow or "gesture" represents, etc.).
- Description of the tour's structure and how it relates to the interests of the intended audiences.
- Annotative cues to maintain coherence and context to offset the effects of "fragmentation" (Thüring *et al.*, 1995).

Hypermedia environment developers could allow both readers and authors to annotate nodes on tours and trails, and facilitate those appearing only in conjunction with that trail. When a trail or tour includes regular documents and nodes, special annotations and labeling could alleviate the following danger. Garzotto, Mainetti and Paolini (1995) note that understanding a node's content sometimes depends on reading other nodes first, or on titles and supplemental links that authors sometimes remove to fit that node into the tour. They give several examples of how including both textual and multimedia nodes out of their normal context could make the contents confusing.

Cuff (1995) proposes a flexible extension of trails, in which the system warns the reader that a link will digress from the trail's theme. The system would also incorporate guided tours, preventing the user from digressing at all. The author could set the *width* of the trail, i.e. how many traversals away from the main trail the reader may digress at any time.

Zellweger (1989) has extended the notion of trails to procedural programmable paths with active entries called *scripts*. This is analogous to a very sophisticated macro facility where the author builds *path macros* (Parunak, 1989) in which entries can perform arbitrary actions ranging from issuing system commands to manipulating variables via an interpreter. Sample actions include playing back a voice annotation, animating a picture or querying a database.

This leads to the school of thought that considers hypermedia as a way of representing and implementing processes. One can represent a process as an associative network which users invoke through navigating. Representing the process steps as nodes and transitions as links enables developers and users to augment the process with

annotations and with links to related information (Scacchi, 1989). Similarly, some developers structure programs as associated networks (e.g. Petri nets) controlling which links to activate based on the user's previous interactions (Stotts & Furuta, 1989).

In a third-generation style authoring environment, authors manually place each entry in the tour list and manually tailor the contents of each tour node. The system could generate previous step, next step, first entry and exit navigation buttons automatically, as well as generating an overview map. A fourth-generation trail and tour authoring environment would use semantic types, keywords and other attributes, access permissions, views and user information to organize and tailor the tour contents dynamically.

On the World Wide Web, the Java tutorial is a guided tour with great depth and comprehensiveness (Sun, 1995). New York University's Information Systems Department Information System (New York University, 1996), designed using the RMM formal hypermedia design methodology (Isakowitz, Stohr & Balasubramanian, 1995), generates guided tours automatically, from a database of course, professor and publication information. Trails (and tours) on the web tend to contain only documents owned by the trail author. Authors need write access to documents to embed "previous" and "next" links and to otherwise annotate and tailor the documents. System developers could provide environments for authors to build trails containing any document by preprocessing each document to append tour commands, supplemental links, annotations, etc. and remove any links the author wishes to filter out for tours, as with Wright and Jones (1997). The environment also might build a trail map which the author could modify. System developers could take advantage of the data model and comprehensive set of navigation commands for implementing traversal within single and multilevel nested guided tours (Garzotto *et al*., 1996). These include starting, suspending, resuming, going to previous and next steps, etc.

Bush foresaw "a new profession of trail blazers, those who find delight in the task of establishing useful trails through the enormous mass of the common record" (Bush, 1945). Given the right tools, the web can provide this massive common record for trail blazers.

3.9. BACKTRACKING AND HISTORY-BASED NAVIGATION

Hypermedia-style interfaces help readers to explore information networks safely and confidently. A reader may be working on a particular task when a link anchor catches the eye and decides to take a *detour*. The reader should be able to select this link knowing that if it is not interesting, or when done exploring that tangent, he or she can return readily to the point of departure and continue the original task. In the insurance scenario, e.g. a potential client may be reading information about homeowners insurance and decide to follow an optional "detour", e.g. a link describing the difference between tenant and homeowners insurance or to a customer testimonial. When done, the client can then backtrack to the main materials. Indeed, the insurance company may wish to take advantage of easy returning to provide several detours that promote their other products, knowing that the customer will not get "lost" by examining these. Bieber and Kacmar (1995) provide examples of backtracking within a geographic information system.

Rosenberg notes three other reasons for backtracking: to review the content of a previously visited node, to recover from a link chosen in error and as part of undoing

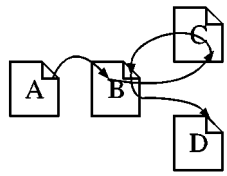

FIGURE 1. A simple hypertext navigation.

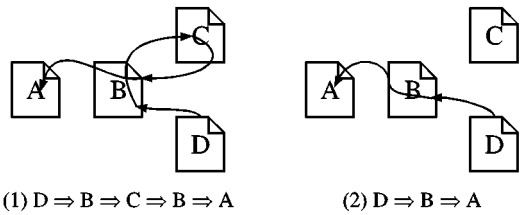

FIGURE 2. Some different backtracking policies.

(Rosenberg, 1996). Backtracking reduces cognitive overhead in that readers know they can always return from a detour or an incorrect traversal. Conceptually, however, backtracking differs from *undoing* in that backtracking returns the reader to a previously visited node in its current state. This could disorient readers if the node contents have changed and appear different than expected. Many web browsers cache node contents, so readers unknowingly may see outdated contents upon backtracking.

Different multiwindow hypermedia systems implement backtracking differently (Bieber & Wan, 1994). Assume the reader uses one of the many hypermedia systems which place each node in a separate window (thus maintaining a level of orientation as the reader can see both the link source and destination). The reader opens node A, traverses a link from node A to node B, traverses a link from node B to node C, clicks on node B and then traverses a different link from node B to node D. Then the user wants to backtrack from node D back to node A. Figure 1 represents this navigation. Different hypermedia systems implement backtracking differently, as Figure 2 shows.

Hypermedia researchers envision full fourth-generation environments in which users will work in multiple hypermedia applications simultaneously (hypermedia email, electronic commerce application, a spreadsheet, a word processor, etc.). Similarly, on the web we envision that users will have multiple windows open in a single editor/browser client and work on several tasks at once. This inspired Bieber and Wan (1994) to distinguish *chronological backtracking* from *task-based backtracking*. Chronological backtracking returns the reader to nodes in the reverse order visited. Task-based backtracking restricts backtracking among nodes within the current logical "task". They propose a simple technique, inferring that the task potentially changes whenever the user opens or clicks on a different window. More sophisticated analysis would infer which groups of windows belong to the same task. Their paper presents a data model and algorithm for each type of backtracking.

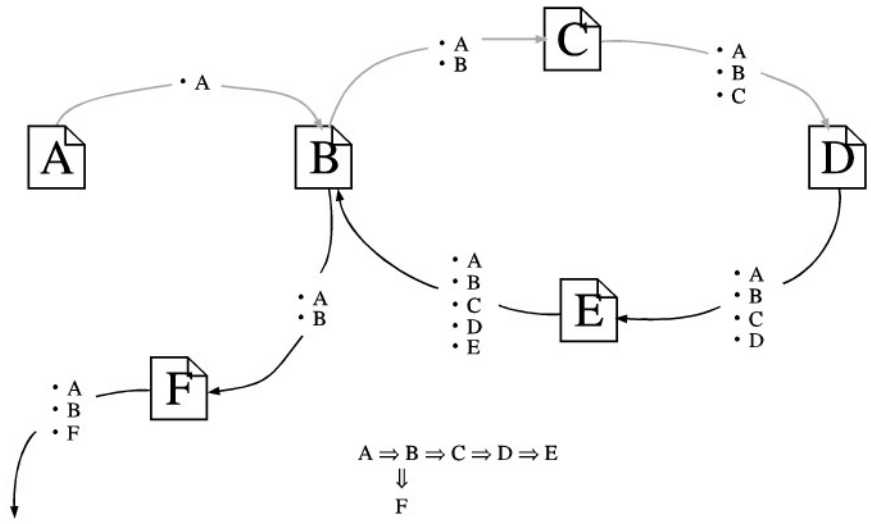

FIGURE 3. How the history path changes during traversal and backtracking.

Garzotto *et al.* (1996) model qualified backtracking. *Parametric backtracking*, written as `go-back(X)`, allows the user to specify a node characteristic in parameter `X`, which causes the system to backtrack to the most recently departed node with that characteristic. *Conditional backtracking*, written as `go-back(query-expression)` evaluates query-expression and returns the reader to the most recently departed node satisfying it.

While backtracking one node at a time eventually brings the reader back to a particular previous node, readers often want to *backjump* directly to a particular departure point. The system should display a session log or *history* list in a clear enough format for the reader to distinguish the node to which he or she wants to return. Often the reader must pick this from a textual list of node title's. Nielsen (1990) augments the node name with the amount of time since the user last visited the node. The Interactive Graphical Documents system represents the history list as a graphical timeline displaying a miniature picture of each node each time it was visited, along with the time it was visited (Feiner, 1985; Utting *et al.*, 1989). The LINKBase system displays a copy of the session log as standard part of each page, so the user always can see how he or she arrived at the current page (SIGLINK, 1994; Balasubramanian, Ma & Yoo, 1995; Balasubramanian, Bieber & Isakowitz, 1997).

Many web browsers implement the history list as a stack augmented with all nodes on the current history path. When the reader backtracks or backjumps to a certain point and then traverses a new link, the browsers remove all nodes on any history paths emanating from that point. For example, in Figure 3, suppose the reader traverses from A to E, and backtracks back to B. At this point, nodes C, D and E will still appear on the history list. When the reader traverses to F, the system removes the history path C to E from the history list.

Nielsen's system, in contrast, treats the history list as a session log. Backtracking adds the nodes onto the list again. When the reader backjumps from the list, the system both

adds the nodes onto the list again and places a checkmark on the backjumped destination to remind the reader of this action when viewing the history list in the future. Still, despite timestamping each entry and these other orientation cues, readers may find a linear session log cannot adequately capture repetitive traversal within a nonlinear network. Implementing the session log as a graphical overview (see Section 3.7) may orient the user better. Authors building guided tours and trails (see Section 3.8) will find a session log especially useful. In systems such as Microcosm (Davis *et al.*, 1992), authors can browse normally to find all nodes and the proper traversal among them, and then package this up as a trail by modifying a copy of the session log.

NoteCards takes an even broader view of the history list, maintaining a full log of all author and reader actions. Users can specify which events to record (node and link creation, traversal, deletion, etc.), making this a tailorable log. While all this information is not necessary for backtracking or jumping, it does provide additional orientation for the user in recalling all steps taken to arrive at the current application state.

Backtracking within a fourth-generation hypermedia environment would allow not only the reader to specify parameter and conditions in an intuitive manner, but would allow an author to set up standard sets of parameters and conditions for a particular application domain. The fourth-generation environment also would support sets of predefined task-based backtracking options tailorable to particular user domains.

Conditional or parametric backtracking would require web browsers to maintain whichever additional parameters that users may wish to query in the history list. These could include those discussed in other sections of this paper, such as semantic link type or document type, title and location or perhaps the parameters passed during traversal. For conditional or parametric backtracking based on document content, the system could search the contents saved in the browser's cache. Creating trails and guided tours will require the system to maintain a supplementary session log. Task-based backtracking will require the system to capture how each document is activated: by opening, clicking, closing the window above it, through backtracking, etc.

3.10. ADDITIONAL HYPERMEDIA FUNCTIONALITY

The papers and presentations at the *Second International Workshop on Incorporating Hypertext Functionality into Software Systems* (Ashman *et al.*, 1996) held in conjunction with the ACM Hypertext '96 conference addressed a wide variety of hypermedia features in software systems. Table 3 classifies these into node, link, navigation and miscellaneous issues.† The present list is by no means complete, but rather gives one an idea of the richness and applications of hypermedia functionality research. Unfortunately, in this paper, we could describe only a small, but important selection of these. In what follows, we briefly describe the ones we left out.

Associative links connect information in some way related, while *annotative links* represent the subset of nodes that comment upon others. While most second-generation links are *hand-crafted*, the larger the information base, the less feasible authors find it to

† We especially wish to thank Carolyn Watters from Acadia University in Canada, a participant in the Hypertext Functionality Workshop II at the Hypertext '96 conference (Ashman *et al.*, 1996), for coming up with the structure of the feature list in Table 3.

TABLE 3
Hypermedia features addressed in the HTF II workshop

A. Node issues
Annotations
Typed nodes
B. Link issues
Associative and annotative links
Computed and hand-crafted links
Dynamically computed and precomputed links
Link attributes
Transclusions
Unidirectional and bidirectional links
C. Navigation issues
Author-created landmarks
Backtracking and interaction history
Browsers and overviews
Content-based information retrieval mechanisms
Different views of a single hypermedia network
Link traversal mechanisms
Node participation in different views
Reader-created bookmarks
Structure-based information retrieval mechanisms
Trails and guided tours
D. Miscellaneous issues
Access to heterogeneous repositories
Automatic link propagation
Hypermedia-aware user-interface controls or widgets
Links to data in any application
Mapping of application objects to hypermedia objects
Standards for content markup and linking

compose all links manually. For domains where the hypermedia network represents applications, which generate display contents at run time (e.g. database management systems, decision support systems, expert systems and geographic information systems), hypermedia support must be *computed* or *mapped* automatically (Bieber, 1992; Bieber & Kimbrough, 1992; Bieber & Kacmar, 1995; Wan, 1996). Also, automated tools such as link apprentices (Bernstein, 1990) can propose lexical links between documents to authors, forming a preliminary tool for fourth generation environments. When *dynamically computed links* can be created periodically instead of the last minute, then they could be *precomputed* to save time when the user wants to traverse them. Without an external linkbase, links are inherently *unidirectional*. Most hypermedia systems support *bidirectional links*, which users can access from either endpoint.

Landmarks are one-way links from everywhere to a specific place, such as a home page. Normally, landmarks appear at the top or bottom of every document, in frames (in browsers such as Netscape) or in menus. Authors create landmarks, whereas readers

create *bookmarks*, which essentially constitute personal landmarks. Web browser hot lists are hypermedia bookmarks. While the concept of hypermedia concerns browsing, many hypermedia systems also incorporate *content-based information retrieval mechanisms.* Readers can use content-based search to locate a relevant starting point for browsing.

Hypermedia structuring and access applies to anything that can be represented as an associative network. As long as the hypermedia engine has some intermediate access or mapping function (Bieber, 1995), we should be able to annotate and access metainformation about objects anywhere, including those in *heterogeneous data, process and knowledge repositories* (Noll & Scacchi, 1996). *Automatic link propagation* concerns link update. When information such as the price of an insurance policy changes, all direct occurrences of that item should be updated, as well as any calculations depending on this price. As part of the Chimera environment, Anderson, Taylor and Whitehead (1994) currently are developing a set of *hypermedia-aware widgets* containing a degree of hypermedia functionality for X-Windows. Any application incorporating these will get this hypermedia functionality for free (Anderson, 1996). We should not only have links to and from HTML documents, but *links to data in any application.* HTML is but one *standard for content markup and linking*. HyTime provides hypermedia support to SGML documents, but also can express locations within documents which are not in HyTime, SGML or HTML (DeRose & Durand, 1994).

4. Looking back and looking ahead

The World Wide Web emerged among a group of globally dispersed physics researchers endeavouring to share information. Hypermedia has grown primarily from computer science and English literature researchers. While both fields build upon interrelationships, hypermedia researchers got a head start on designing the high-level features we discuss. Yet just as web environments can benefit greatly from hypermedia functionality, hypermedia researchers are learning tremendously about distribution and scale (Engelbart, 1990; Malcolm *et al*., 1991; Parunak, 1991*b*) from the web community.† The World Wide Web is a wonderful delivery vehicle for hypermedia support. We urge each field to embrace and prosper from the successes of the other.

Returning of the list of challenges posed in the insurance scenario that opened Section 3, we conclude that applications can leverage the Web's hypermedia infrastructure to improve effectiveness in many ways. The high-level structuring, navigation and annotation features we discuss help authors organize information so that users can access it directly while maintaining context and orientation. Applications could employ annotations and semantically typed templates to create fourth-generation environments for collaborative authoring. These authoring environments and the information they produce both can be customized for specific uses and users. Transclusions, warm and hot linking and versioning can support this customization as well as information update and access to historical information. Access privileges extend security to hypermedia-structured information.

† Smith, Newman and Parks (1997, this issue), e.g. conclude that hypertext usability research cannot be utilized as the basis for web usability research because of the inherent differences caused by scale between primarily small hypertext systems and large, distributed web applications.

In summary, application developers should strongly consider hypermedia structuring and navigation for organizing, merging and giving users flexible access within large bodies of information. But they will do so *en masse* only when the appropriate tools exist. We call on web developers to provide high-level hypermedia features and their respective sophisticated authoring environments so that every application developer, author and reader can take advantage of them.

We wish to thank the participants of the Hypertext Functionality Workshops I and II, held in conjunction with the European Conference on Hypermedia Technologies (ECHT) 1994 and the Hypertext'96 Conference. Special thanks to Keith Andrews, Alejandra Garrido, Paul Kahn and Hermann Maurer for reviewing drafts of this paper. This research has been supported generously by the NASA JOVE faculty fellowship program, by the New Jersey Center for Multimedia Research, by the New Jersey Commission on Science and Technology, by the New Jersey Institute of Technology (NJIT) under grant #991967, by the National Center for Transportation and Industrial Productivity at NJIT, by the New Jersey Department of Transportation and by a grant from the AT&T Foundation. The authors also may be reached through their URLs: http://megahertz.njit.edu/ ~bieber; http://www.cs.unibo.it/ ~fabio; http://www.e-papyrus.com/personal/bala.html; http://rieska.oulu.fi/ ~ hok.

References

AKSCYN, R., MCCRACKEN, D. & YODER, E. (1988). KMS: a distributed hypermedia system for managing knowledge in organizations. *Communications of the ACM*, **31**(7)**,** 820–835.

ANDERSON, K. (1996). Providing automatic support for extra-application functionality. In H. L. ASHMAN, V. BALASUBRAMANIAN, M. BIEBER & H. OINAS-KUKKONEN, Eds. *Proceedings of the 2nd International Workshop on Incorporating Hypertext Functionality into Software Systems (HIF II) Hypertext '96 Conference.* [http://space.njit.edu:5080/HTFII/Proceedings.html].

ANDERSON, K. M., TAYLOR, R. N. & WHITEHEAD, E. J. Jr (1994). Chimera: hypertext for heterogeneous software environments. *Proceedings of the European Conference on Hypermedia Technologies*, pp. 94–107, New York, NY: ACM Press.

ANDREWS, K. (1996). Applying hypermedia research to the World Wide Web. *Workshop on Hypermedia Research and the World Wide Web*, *Hypertext '96 Conference*, Washington. [http://www. iicm.edu/apphrweb].

ASHMAN, H. L., BALASUBRAMANIAN, V., BIEBER, M. & OINAS-KUKKONEN, H., Eds. (1996). *Proceedings of the 2nd International Workshop on Incorporating Hypertext Functionality into Software Systems (HTF II)*, *Hypertext '96 Conference*, *Washington.* [http://space.njit.edu: 5080/ HTFII/Proceedings.html].

ASHMAN, H. L., BALASUBRAMANIAN, V., BIEBER, M. & OINAS-KUKKONEN, H., Eds. (1997). *Proceedings of the 3rd International Workshop on Incorporating Hypertext Functionality into Software Systems (HTF/III)*, *Hypertext '97 Conference.* [http://space.njit.edu:5080/ HTFI II/].

BALASUBRAMANIAN, V., BASHIAN, A. & PORCHER, D. (1997). A large-scale hypermedia application using document management and web technologies. *Hypertext '97 Proceedings*, pp. 134–145, New York, NY: ACM Press.

BALASUBRAMANIAN, V., BIEBER, M. & ISAKOWITZ, T. (1997). Systematic hypermedia design. Technical Report, CIS Department, New Jersey Institute of Technology, Newark, NJ 07102-1982.

BALASUBRAMANIAN, V., MA, B. & YOO, J. (1985). A systematic approach to designing a WWW application. *Communications of the ACM*, **38**(8)**,** 47–48.

BENYON, D., STONE, D. & WOODROFFE, M. (1997, this issue). Experience with developing multimedia courseware for the WWW: the need for better tools. In S. BUCKINGHAM SHUM & C. MCKNIGHT, Eds. Special Issue of *International Journal of Human–Computer Studies.*

BERNERS-LEE, T., CAILLIAU, R., LUOTONEN, A., NIELSEN, H. F. & SECRET, A. (1994). The World Wide Web. *Communications of the ACM*, **37**(8)**,** 76–82.

BERNSTEIN, M. (1990). An apprentice that discovers hypertext links. Hypertext: concepts, systems and applications. *Proceedings of the European Conference on Hypertext*, pp. 212–223. Cambridge, UK: Cambridge University Press.

BIEBER, M. (1992). Automating hypermedia for decision support. *Hypermedia*, **4,** 83–110. [http://space.njit.edu:5081/bieber/~pub/Bi92.html].

BIEBER, M. (1995). On integrating hypermedia into decision support and other information systems. *Decision Support Systems*, **14,** 251–267. [http://space.njit.edu:5081/bieber/~pub/Bi95.html].

BIEBER, M. (1996). What every developer should know about hypertext. *Proceedings of the Americas Conference of the Association of Information Systems.* [http://space.njit.edu:5081/bieber/~pub/Bi96.html].

BIEBER, M. (1997). Advancing information comprehension through hypertext. In J. MAYFIELD & C. NICHOLAS, Eds. *Advances in Intelligent Hypertext*. Berlin: Springer. [http://space.njit.edu:5081/bieber/ ~ pub/Bi97.html].

BIEBER, M. & ISAKOWITZ, T., Eds. (1995). Designing hypermedia applications (special issue). *Communications of the ACM*, **38**(8) [http://space.njit.edu:5080/cacm/overview.html].

BIEBER, M. & KACMAR, C. (1995). Designing hypertext support for computational applications. *Communications of the ACM*, **38**(8)**,** 99–107. [http://space.njit.edu:5081/bieber/~pub/BK95.html; http://space.njit.edu:5080/cacm/overview.html].

BIEBER, M. & KIMBROUGH, S. O. (1992). On generalizing the concept of hypertext. *Management Information System Quarterly*, **16**(1)**,** 77–93. [http://space.njit.edu:5081/bieber/~pub/BK92.html].

BIEBER, M. & VITALI, F. (1997). Toward support for hypermedia on the World Wide Web. *IEEE Computer*, **30**(1)**,** 62–70. [http://space.njit.edu:5081/bieber/~pub/BV97.html].

BIEBER, M. & WAN, J. (1994). Backtracking in a multiple window hypertext environment. *Proceedings of the European Conference on Hypermedia Technologies*, pp. 158–166. New York, NY: ACM Press. [http://space.njit.edu:5081/bieber/~pub/BW94.html].

BROWN COMPUTER GRAPHICS GROUP (1995). A celebration of Vannevar Bush's 1945 vision. [http://www.cs.brown.edu/research/graphics/html/info/vannevar_bush.html].

BURGER, A. M., MEYER, B. D., JUNG, C. P. & LONG, K. B. (1991). The virtual notebook system. *Hypertext '91 Proceedings*, pp. 395–402. New York, NY: ACM Press.

BUSH, V. (1945). As we may think. *Atlantic Monthly*, **176,** 101–108. [http://www.isg.sfu.ca/~duchier/misc/vbush/].

CARR, L., DEROURE, D., HALL, W. & HILL, G. (1995). The distributed link service: a tool for publishers, authors and readers. *Proceedings of the 4th International WWW Conference*. [http://wwwcosm.ecs.soton.ac.uk/dls/link_service.html].

CATLEDGE, L. & PITKOW, J. (1995). Characterising browsing strategies in the World Wide Web. *Proceedings of the 3rd International World Wide Web Conference*, Darmstadt, Germany. [http://www.igd.fhg.de/www/www95/papers/80/userpatterns/UserPatterns.Paper4.formatted.html].

CATLIN, K. S., GARRET, L. N. & LAUNHARDT, J. A. (1991). Hypermedia templates: an author's tool. *Hypertext '91 Proceedings*, pp. 147–160. New York, NY: ACM Press.

CATLIN, T., BUSH, P. & YANKELOVICH, N. (1989). InterNote: extending a hypermedia framework to support annotative collaboration. *Hypertext '89 Proceedings*, pp. 365–378. New York, NY: ACM Press.

COLLIER, G. H. (1987). Thoth II: hypertext with explicit semantics. *Hypertext '87 Proceedings*, pp. 269–290. New York, NY: ACM Press.

CONKLIN, J. (1987). Hypertext: an introduction and survey. *IEEE Computer*, **20**(7)**,** 17–41.

CONKLIN, E. J. & BEGEMAN, M. L. (1989). gIBIS: a tool for all reasons. *Journal of the American Society for Information Science*, **40**(3)**,** 200–213.

GROUP DECISION SUPPORT SYSTEMS, INC. (1996). *Tools for Team Collaboration.* [http://www.gdss.com].

CUFF, L. D. (1995). Commercial hypertext publishing: electronic books using trails and the author–publisher–reader model. *World Wide Web Journal*, **1**. [http://www.w3.org/pub/WWW/Journal/1/i.309/paper/309.html].

DAVIS, H. C. (1995*a*). To embed or not to embed. *Communications of the ACM*, **38**(8), 108–109.

DAVIS, H. C. (1995*b*). *Data integrity problems in an open hypermedia link service*. Ph.D. Dissertation, University of Southampton.

DAVIS, H. C., HALL, W., HEATH, I., HILL, G. & WILKINS, R. (1992). Towards an integrated information environment with open hypermedia systems. *Proceeding of the ACM Conference on Hypertext*, pp. 181–190. New York, NY: ACM Press.

DAVIS, H. C., KNIGHT, S. & HALL, W. (1994). Light hypermedia link services: a study of third-party application integration. *Proceedings of the European Conference on Hypermedia Technologies*, pp. 41–50. New York, NY: ACM Press.

DEROSE, S. J. (1989). Expanding the notion of links. *Hypertext '89 Proceedings*, pp. 249–257. New York, NY: ACM Press.

DEROSE, S. J. & DURAND, D. G. (1994). *Making Hypermedia Work, A User's Guide to HyTime*. Dordrecht: Kluwer Academic Publishers.

ENGELBART, D. C. (1990). Knowledge domain interoperability and an open hyperdocument system. *Proceedings of the Conference on Computer-Supported Cooperative Work*, pp. 143–156. New York, NY: ACM Press.

ENGELBART, D. C. & ENGLISH, W. (1968). A research center for augmenting human intellect. *AFIPS Conference Proceedings, Fall Joint Computer Conference*, Vol. 33, pp. 395–410.

FEINER, S. (1985). Interactive documents. In P. WHITNEY & C. KENT, Eds. *Design in the Information Environment*. pp. 118–132. New York, NY: Alfred Knopf.

GARZOTTO, F., MAINETTI, L. & PAOLINI, P. (1995). Hypertext design, analysis, and evaluation issues. *Communications of the ACM*, **38**(8), 74–86.

GARZOTTO, F., MAINETTI, L. & PAOLINI, P. (1996). Navigation in hypermedia applications: modeling and semantics. *Journal of Organizational Computing and Electronic Commerce*, **6**(3), 211–237.

HALASZ, F. G. (1988). Reflections on NoteCards: seven issues for the next generation of hypermedia systems. *Communications of the ACM*, **31**(7), 836–855.

HALASZ, F. & SCHWARTZ, M. (1994). The Dexter hypertext reference model. *Communications of the ACM*, **37**(2), 30–39.

HILTZ, S. R. (1993). *The Virtual Classroom: Learning without Limits via Computer Networks*. Human Computer Interaction Series, Norwood, NJ: Ablex Publishing Corp.

HILTZ, S. R. & TUROFF, M. (1993). *The Network Nation*. Cambridge, MA: MIT Press.

GRAMLICH, W. C. (1994). Public annotation systems. [http://playground.sun.com:80/~gramlich/1994/annote/].

INTERNATIONAL STANDARDS ORGANIZATION (1986). ISO 8879, Information Processing—Text and Office Systems—Standard Generalized Markup Language (SGML).

ISAKOWITZ, T., STOHR, E. & BALASUBRAMANIAN, P. (1995). RMM: a methodology for structuring hypermedia design. *Communications of the ACM*, **38**(8), 34–44.

JESSUP, L. M. & VALACICH, J. S. (1993). *Group Support Systems: New Perspectives*. New York, NY: Macmillan.

JOYCE, M. (1991). Storyspace as a hypertext system for writers and readers of varying ability. *Hypertext '91 Proceedings*, pp. 381–388. New York, NY: ACM Press.

KACMAR, C. (1996). A process approach to providing hypermedia services to existing non-hypermedia applications. *Electronic Publishing: Organization, Dissemination, and Distribution*, **8**(1), 31–48.

KAINDL, H. & SNAPRUD, M. (1991). Hypertext and structured object representation: a unifying view. *Hypertext '91 Proceedings*, pp. 345–358. New York, NY: ACM Press.

KAPPE, F. (1995). Maintaining link consistency in distributed hyperwebs. *Proceedings of INET '95*. Internet Society. [http://info.isoc.org:80/HMP/PAPER/073/].

LA LIBERTE, D. (1994). A protocol for scalable group and public annotations. [http://union.ncsa.uiuc.edu/~liberte/www/scalable-annotations.html].

LEE, Y. K., YOO, S. -J., YOON, K. & BERRA, P. B. (1996). Querying structured hyperdocuments. *Proceedings of the 29th Annual Hawaii International Conference on System Sciences*, Vol. II, pp. 155–164. Los Alamitos, CA: IEEE Computer Society Press.

LEIDNER, D. E. & JARVENPAA, S. L. (1995). The use of information technology to enhance management school education: a theoretical view. *Management Information Systems Quarterly*, **19**(3), 265–291.

LUCARELLA, D. (1990). A model for hypertext-based information retrieval. Hypertext: concepts, systems and applications. *Proceedings of the European Conference on Hypertext*, pp. 81–94. Cambridge, UK: Cambridge University Press.

MAIOLI, C., SOLA, S. & VITALI, F. (1993). Wide-area distribution issues in hypertext systems. *ACM SIGDOC '93 Conference Proceedings*, pp. 185–198. New York, NY: ACM Press.

MALCOLM, K. C., POLTROCK, S. E. & SCHULER, D. (1991). Industrial strength hypermedia: requirements for a large engineering enterprise. *Hypertext '91 Proceedings*, pp. 13–24. New York, NY: ACM Press.

MALONEY, M. (1995). *Hypertext links in HTML*. W3C Working Group Discussion Paper [http://www.sq.com/papers/Relationships.html].

MARSHALL, C. C., HALASZ, F. G., ROGERS, R. A. & JANSSEN, W. C. Jr (1991). Aquanet: a hypertext tool to hold your knowledge in place. *Hypertext '91 Proceedings*, pp. 261–275. New York, NY: ACM Press.

MARSHALL, C. C. & IRISH, P. M. (1989). Guided tours and on-line presentations: how authors make existing hypertext intelligible for readers. *Hypertext '89 Proceedings*, pp. 15–42. New York, NY: ACM Press.

MARSHALL, C. C. & SHIPMAN, F. III (1995). Spatial hypertext: designing for change. *Communications of the ACM*, **38**(8), 88–97.

MAURER, H. (1996). *HyperWave: The Next Generation Web Solution*. London: Addison-Wesley. [http://www.iicm.edu/hgbook].

MEYROWITZ, N. (1989). Hypertext—does it reduce cholesterol, too? Hypertext '89 Keynote Address, IRIS Technical Report 89-9, Brown University, Providence, RI.

MONNARD, J. & PASQUIER-BOLTUCK, J. (1992). An object-oriented scripting environment for the WEBSs electronic book system. *Proceeding of the ACM Conference on Hypertext*, pp. 81–90. New York, NY: ACM Press.

MYLONAS, E. & HEATH, S. (1990). Hypertext from the data point of view: paths and links in the Perseus project. Hypertext: concepts, systems and applications. *Proceedings of the European Conference on Hypertext*, pp. 324–336. Cambridge, UK: Cambridge University Press.

NELSON, T. H. (1987). *Literary Machines 87.1*. Sausalito, CA: Sausalito Press.

NELSON, T. H. (1988). Managing immense storage. *BYTE*, **13**, 225–238.

NELSON, T. H. (1995). The heart of connection: hypermedia unified by transclusion. *Communications of the ACM*, **38**(8), 31–33.

NIELSEN, J. (1990). The art of navigating through hypertext. *Communications of the ACM*, **33**(3), 287–310.

NANARD, J. & NANARD, M. (1995). Hypertext design environments and the hypertext design process. *Communications of the ACM*, **38**(8), 49–56.

NATIONAL CENTER FOR SUPERCOMPUTING APPLICATIONS (UNIVERSITY OF ILLINOIS) (1995). The NCSA Mosaic Homepage. [http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSAMosaicHome.html].

NEW YORK UNIVERSITY (1996). Department of Information Systems home page. [http://is-2.stern.nyu.edu/isweb].

NEWCOMB, S., KIPP, N. & NEWCOMB, V. (1991). The 'HyTime' *hypermedia*/time-based document structuring language. *Communications of the ACM*, **34**(11), 67–83.

NOLL, J. & SCACCHI, W. (1996). Repository support for the virtual software enterprise. In H. L. ASHMAN, V. BALASUBRAMANIAN, M. BIEBER & H. OINAS-KUKKONEN, Eds. *Proceedings of the 2nd International Workshop on Incorporating Hypertext Functionality into Software Systems (HTF II), Hypertext '96 Conference*. [http://space.njit.edu:5080/HTFII/[Proceedings.html].

NUNAMAKER, J. F., DENNIS, A. R., VALACICH, J. S., VOGEL, D. R. & GEORGE, J. F. (1991). Electronic meeting systems to support group work. *Communications of the ACM*, **34**(7), 41–61.

OINAS-KUKKONEN, H. (1996). Debate browser—an argumentation tool for MetaEdit+environment. *Proceedings of the 7th European Workshop on Next Generation of CASE Tools (NGCT '96)*, Crete, Greece, pp. 77–86. Trondheim, Norway: Norwegian University of Science and Technology.

OINAS-KUKKONEN, H. (1997*a*). Embedding hypermedia into information systems. *Proceedings of the 30th Annual Hawaii International Conference on System Science*, Vol. VI, pp. 187–196. Los Alamitos: IEEE Computer Society Press.

OINAS-KUKKONEN, H. (1997*b*). Towards greater flexibility in software design systems through hypermedia functionality. *Journal of Information and Software Technology* (forthcoming).

ØSTERBYE, K. & WIIL, U. K. (1996). The flag taxonomy of open hypermedia systems. *Hypertext '96 Proceedings*, pp. 129–139. New York, NY: ACM Press.

PARUNAK, H. V. D. (1989). Hypermedia topologies and user navigation. *Hypertext '89 Proceedings*, pp. 43–50. New York, NY: ACM Press.

PARUNAK, H. V. D. (1991*a*). Ordering the information graph. In E. BERK & J. DEVLIN, Eds. *Hypertext/Hypermedia Handbook*, pp. 299–325. New York, NY: Intertext Publications/McGraw-Hill.

PARUNAK, H. V. D. (1991*b*). Toward industrial strength hypermedia. In E. BERK & J. DEVLIN, Eds. *Hypertext/Hypermedia Handbook*, pp. 381–395. New York, NY: Intertext Publications/McGraw-Hill.

PITKOW, J. & KEHOE, C. (1995). GVU*'s 4th WWW User Survey Home Page.* [http://www.cc.gatech.edu/gvu/user_surveys/survey-10-1995/].

RADA, R. (1990). Hypertext writing and document reuse: the role of a semantic net. *Electronic Publishing: Organization, Dissemination and Distribution*, **3**, 3–13.

RANA, A. & BIEBER, M. (1997). Collaborative hypermedia education framework. *Proceedings of the 30th Annual Hawaii International Conference on System Science*, Vol. II, pp. 610–619. Los Alamitos: IEEE Computer Society Press. [http://space.njit.edu:5081/bieber/~pub/RB97.html].

RAO, U. & TUROFF, M. (1990). Hypertext functionality: a theoretical framework. *International Journal of Human–Computer Interaction*, **2**(4), 333–358.

RIZK, A. & SAUTER, L. (1992). Multicard: an open hypermedia system. *Proceeding of the ACM Conference on Hypertext*, pp. 4–10. New York, NY: ACM Press.

ROSENBERG, J. (1996). The structure of hypertext activity. *Hypertext '96 Proceedings*, pp. 22–30. New York, NY: ACM Press.

SCACCHI, W. (1989). On the power of domain-specific hypertext environments. *Journal of the American Society for Information Science*, **40**(3), 183–191.

SCHLOSS, R. J. (1996). Novel business uses of independently created hyperlinks in the World Wide Web: basic mechanism and examples. *Proceedings of the 29th Annual Hawaii International Conference on System Science*, Vol. II, pp. 137–146. Los Alamitos: IEEE Computer Society Press.

SCHWABE, D., ROSSI, G. & BARBOSA, S. D. J. (1996). Systematic hypermedia application design with OOHDM. *Hypertext '96 Proceedings*, pp. 116–128. New York, NY: ACM Press.

SHACKELFORD, D. E., SMITH, J. B. & SMITH, F. D. (1993). The architecture and implementation of a distributed hypermedia storage system. *Hypertext '93 Proceedings*, pp. 1–13. New York, NY: ACM Press.

SIGLINK (1994). SIGLINK Home Page. [http://www.acm.org/siglink/].

SMITH, P., NEWMAN, I. & PARKS, L. (1997, this issue). Virtual hierarchies and virtual networks: some lessons from hypermedia usability research applied to the World Wide Web. In S. BUCKINGHAM SHUM & C. MCKNIGHT, Eds. *International Journal of Human-Computer Studies* (special issue).

SIMPSON, R., RENEAR, A., MYLONAS, E. & VAN DAM, A. (1996). 50 Years After 'As We May Think': The Brown/MIT Vannevar Bush Symposium. Interactions, **3**(2), 47–67. [http://www.cs.brown.edu/memex/].

STEIN, E. W. & ZWASS, V. (1995). Actualizing organizational memory with information technology. *Information Systems Research*, **6**(2), 85–117.

STOTTS, P. D. & FURUTA, R. (1989). Petri-net-based hypertext: document structure with browsing semantics. *ACM Transactions on Information Systems*, **7**(1), 3–29.

STREITZ, N., HAAKE, J., HANNEMANN, J., LEMKE, A., SCHULER, W., SCHÜTT, H. & THÜRING, M. (1992). SEPIA: a cooperative hypermedia authoring environment. *Proceeding of the ACM Conference on Hypertext*, pp. 11–22. New York, NY: ACM Press.

SUN (1995). Sun Microsystems, Inc. Java API Documentation. [http://www.javasoft.com/JDK-1.0/api/packages.html].

SUN (1996). Sun Microsystems, Inc. Jeeves, Coming to Serve You! [http://www.javasoft.com/jeeves].

TANIK, M., YEH, R., BIEBER, M., KURFESS, F., LIU, Q., MCHUGH, J., RANA, A., ROSSAK, W. & NG, P. (1996). Issues and architecture for electronic enterprise engineering (EEE). *Proceedings of the 2nd World Conference on Integrated Design and Process Technology*, Vol. 2, pp. 57–62. Austin: Society for Design and Process Science, [http://www.cs.njit.edu/~eee/pubs/sdps96.html].

TAUSCHER, L. & GREENBERG, S. (1997, this issue). How people revisit Web pages: Empirical findings and implications for the design of history systems. In S. BUCKINGHAM SHUM & C. MCKNIGHT, Eds. *International Journal of Human–Computer Studies* (special issue).

TERVONEN, I. & OINAS-KUKKONEN, H. (1996). Reorganized inspection process: problems encountered and resolved. *Software Process—Improvement and Practice*, **2**(2), 97–110.

TERVONEN, I., KEROLA, P. & OINAS-KUKKONEN, H. (1997). An organizational memory for quality-based software design and inspection: A collaborative multiview approach with hyperlinking capabilities. *Proceedings of the 30th Annual Hawaii International Conference on System Science*, Vol. II, pp. 290–299. Los Alamitos: IEEE Computer Society Press.

THIMBLEBY, H. (1997, this issue). Gentler: a tool for systematic web authoring. In S. BUCKINGHAM SHUM & C. MCKNIGHT, Eds. *International Journal of Human–Computer Studies* (special issue).

THÜRING, M., HANNEMANN, J. & HAAKE, J. M. (1995). Designing for comprehension: a cognitive approach to hypermedia development. *Communications of the ACM*, **38**(8), 57–66.

TRIGG, R. H. (1988). Guided tours and tabletops: tools for communicating in a hypertext environment. *ACM Transactions on Office Information Systems*, **6**(4), 398–414.

TRIGG, R. H. & WEISER, M. (1986). TextNet: A network-based approach to text handling. *ACM Transactions on Office Information Systems*, **4**(1), 1–23.

TUROFF, M. (1991). Computer mediated communication requirements for group support. *Journal of Organizational Computing*, **1**(1), 85–113.

TUROFF, M. & HILTZ, S. R. (1995). Computer based Delphi processes. In M. ADLER & E. ZIGLIO, Eds. *Gazing into the Oracle: The Delphi Method and its Application to Social Policy and Public Health*, pp. 56–88. London: Kingsley Publishers.

THE URN IMPLEMENTORS (1996). Uniform resource names. A progress report. *D-Lib Magazine*, ISSN 1082-9873. [http://www.dlib.org/dlib/february96/02arms.html].

UTTING, K. & YANKELOVICH, N. (1989). Context and orientation in hypermedia networks. *ACM Transactions on Information Systems*, **7**(1), 58–84.

VANZYL, A. J., CESNIK, B., HEATH, I. & DAVIS, H. C. (1994). Open hypertext systems: an examination of requirements, and analysis of implementation strategies, comparing Microcosm, HyperTED, and the World Wide Web. [http://elecpress.lib.monash.edu.au/papers/openhypermedia.html].

VERBYLA, J. L. M. & ASHMAN, H. L. (1994). A user-configurable hypermedia-based interface via the functional model of the link. *Hypermedia*, **6**(3), 193–208.

VITALI, F., CHIU, C. & BIEBER, M. (1997). Extending HTML in a principled way with displets. *Computer Networks and ISDN Systems*, Elsevier (forthcoming). [http://space.njit. edu:5081/bieber/ ~ pub/VCB97.html].

VITALI, F. & DURAND, D. (1995). Using versioning to provide collaboration on the WWW. *Proceedings of the 4th International World Wide Web Conference*, Boston. *World Wide Web Journal*, **1**, 37–50. [http://www.w3.org/pub/WWW/Journal/1/vitali.190/paper/190.html].

WAN, J. (1996). *Integrating hypertext into information systems through dynamic linking*. Ph.D. Dissertation, New Jersey Institute of Technology, Institute for Integrated Systems Research, Newark, NJ 07102.

WANG, W. & RADA, R. (1995). Experiences with semantic net based hypermedia. *International Journal of Human–Computer Studies*, **43,** 419–439.

WIIL, U.K. Ed. (1997). *Proceedings of the 3rd Workshop on Open Hypermedia Systems, Hypertext '97 Conference*. Technical Report, Danish National Centre for IT Research, Aarhus University. [http://www.daimi.aau.dk/~kock/OHS-HT97/].

WORLD WIDE WEB CONSORTIUM (1995). HTTP—Hypertext Transfer Protocol. [http://www.w3.org/pub/WWW/Protocols/].

WORLD WIDE WEB CONSORTIUM (1996). Part 1 (Addressing). *World Wide Web Journal*, **2,** [http://www.w3.org/pub/WWW/Journal/2DRAFT/].

WRIGHT, C. & JONES, R. (1997). *Computer Networks*. [http://idun.unl.ac.uk/~ex14jonesr/soc/units/co210/co210.html].

WURMAN, R. S. (1989). *Information Anxiety*. New York, NY: Doubleday.

YANKELOVICH, N., Ed. (1988). *Hypertext on Hypertext. Electronic HyperCard version for the Macintosh*. New York, NY: ACM Press.

YANKELOVICH, N., HAAN, B., MEYROWITZ, N. & DRUCKER, S. (1988). Intermedia: the concept and the construction of a seamless information environment. *IEEE Computer*, **21**(1)**,** 81–96.

YODER, E., AKSCYN, R. & MCCRACKEN, D. (1989). Collaboration in KMS, a shared hypermedia system. *CHI '89 Proceedings*, pp. 37–42. New York, NY: ACM Press.

ZELLWEGER, P. (1989). Scripted documents: a hypermedia path mechanism. *Hypertext '89 Proceedings*, pp. 1–14. New York, NY: ACM Press.