

***Final Report of the NSF  
Workshop on Hyperbase Systems***

*Editors*

John J. Leggett  
John L. Schnase  
John B. Smith  
Edward A. Fox

July 1993  
TAMU-HRL 93-002

---

## Workshop Participants

---

Christopher Clifton	Northwestern University
W. Bruce Croft	University of Massachusetts
Edward A. Fox	Virginia Polytechnic Institute and State University
Mark E. Frisse	Washington University School of Medicine
Richard K. Furuta	Texas A&M University
Anja Haake	GMD - IPSI
Hector J. Hernández	New Mexico State University
David L. Hicks	Texas A&M University
Charles J. Kacmar	Florida State University
Danny B. Lange	Technical University of Denmark & Tokyo University
John J. Leggett	Texas A&M University
Raymond McCall	University of Colorado
Don McCracken	Knowledge Systems, Inc.
Steven E. Poltrock	Boeing Computer Services
Laurence C. Rosenberg	National Science Foundation
John L. Schnase	Washington University School of Medicine
Helge Schütt	Stürtz Electronic Publishing GmbH
Douglas E. Shackelford	University of North Carolina
John B. Smith	University of North Carolina
P. David Stotts	University of North Carolina
Uffe Kock Wiil	University of Aalborg
Maria Zemankova	National Science Foundation
Kasper Østerbye	University of Aalborg

# Final Report of the NSF Workshop on Hyperbase Systems

## *Editors*

John J. Leggett  
John L. Schnase  
John B. Smith  
Edward A. Fox

*This workshop was supported by grant IRI-9217185 from the Information Technology & Organizations Program and the Database & Expert Systems Program of the Information, Robotics, & Intelligent Systems Division of the Computer & Information Science & Engineering Directorate of the National Science Foundation. Any opinions, findings, conclusions, or recommendations expressed in this report are those of the workshop participants and do not necessarily reflect the views of the National Science Foundation.*

TAMU-HRL 93-002  
Hypermedia Research Laboratory  
Department of Computer Science  
Texas A&M University  
College Station, Texas 77843-3112  
[hrl@bush.cs.tamu.edu](mailto:hrl@bush.cs.tamu.edu)

July 1993

# Table of Contents

<b>1. INTRODUCTION</b>	1
	2
<b>2. BASIC CONCEPTS AND TERMINOLOGY</b>	2
<b>3. CONTEXT OF THE WORKSHOP</b>	3
<b>4. APPLICATIONS OF HYPERBASE SYSTEMS</b>	4
<b>5. KEY AREAS OF HYPERBASE SYSTEMS RESEARCH</b>	6
5.2 Node, Link and Structure Management	11
5.3 Integration of Information Retrieval	12
5.4 Version Control	15
5.4.2 Summary of Discussion	18
5.5 Concurrency Control, Transaction Management and Notification Control	18
<b>6. CONCLUSIONS</b>	19
<b>APPENDIX I - WORKSHOP PARTICIPANT ADDRESSES</b>	29
APPENDIX I - WORKSHOP PARTICIPANT ADDRESSES (continued)	30
<b>APPENDIX II - LIST OF INVITED PRESENTATIONS</b>	31

## **WORKSHOP ON HYPERBASE SYSTEMS**

### **Executive summary**

This report presents the results of an NSF-sponsored workshop on hyperbase systems. The workshop brought together academic and industrial researchers from the areas of hypermedia, database, information retrieval and collaborative systems for an intensive two-day workshop. The workshop was held on October 15 and 16, 1992, in Washington, D.C. The hyperbase system architectures discussed at the workshop can be characterized as large-scale, open, distributed, heterogeneous and extensible, with capabilities for computation and collaboration. Discussion of typical application environments included digital libraries, large-scale collaborative systems, medical information systems and large engineering enterprises. The workshop concentrated on hyperbase systems and did not include hypermedia environments or user-interface issues, except as they might impact the hyperbase. The participants considered the following key areas of hyperbase systems research: models and architectures; node, link and structure management; integration of information retrieval techniques; version control; and concurrency control, transaction management and notification control.

Hyperbase research is at an early stage of development, and, as a result, the workshop raised many more questions than it answered. However, the goal of fostering cross fertilization of ideas from diverse and interdisciplinary communities has been achieved. Participants discovered common research interests and a substantial overlap in issues and technical requirements for their systems. There is a consensus that the greatest opportunity to influence the field will occur in the next few years. A coordinated, multidisciplinary research program is needed to define fundamental principles of hyperbase systems. This report takes a first step by identifying critical research issues facing hyperbase system researchers, current progress on the issues, potential methods of approach and a general research agenda for the field.

## 1. INTRODUCTION

Advanced information management systems of the future will not resemble the systems of today. Ubiquitous access to high-capacity computer networks will lead to massive data and process distribution, and entirely new forms of collaborative work will emerge. New media and data types will revolutionize computing environments, requiring petabyte or greater storage capabilities. Even today, we see specifications for new information management systems that call for terabyte storage of multimedia data and widespread network access. For example, initial requirements for the Bush Presidential Library information system call for storage of 300,000 image scanned documents, 6000 hours of color video and audio, 3 million photographs and hundreds of thousands of separate polls. Millions of additional documents are expected to be added to the library over its lifetime. The Bush Library will be the nation's first digital presidential library and will provide access to all of the above information through local, national and global networks. The storage management requirements for advanced information management applications, such as those for the Bush Library, overwhelm existing theory and practice, and there is a critical need for innovative new solutions [19, 97].

Hypermedia offers a promising new approach to very large scale information management [41, 77]. Hypermedia applications have traditionally been designed and implemented for single user/single machine use. The hyperbases they have managed have been small, and data management has been provided directly by the underlying operating system's file system or a commercial database system. Recently, however, several research prototypes have expanded the traditional hypermedia system architecture by including a separable hyperbase system. These research prototypes draw upon and extend techniques that have emerged from database, information retrieval, distributed, operating and collaborative systems research.

Advanced information management systems, including hypermedia and collaborative systems, require management of data from a very rich and complex set of data types as well as dynamic management of the structure of the data. Hyperbase systems must support not only storage and manipulation of complex multimedia data but also storage and manipulation of connectivity data. In addition to the usual requirements for permanence of data, controlled sharing, backup, and recovery, hyperbase systems must be capable of modeling complex interrelationships and providing direct support for novel data types. Hyperbase systems must also be able to handle lengthy transactions, transactions unique to hypermedia, massive distribution of functionality, extensibility, notification, versioning of hypermedia data and structure and scaling to the hundreds of terabytes.

We are currently at a crucial point in the development of hyperbase systems. The need for high quality, directed research on hyperbase systems has become very apparent to researchers in many fields including the hypermedia, database, information retrieval and collaborative systems areas. Although several researchers have designed new data models and architectures for hyperbase systems, the field has reached consensus on neither. We need to discover the theoretical foundations for the representation and manipulation of hyperbases, and we must define meaningful metrics for comparison of proposed models and architectures of hyperbase systems. At this time, the most fundamental issues are not well understood, and there is no commonly accepted formal description of hyperbases or hyperbase systems. In order to fully realize the potential of hypermedia and make it useful in real-world settings, basic research on hyperbase systems is critically needed, and fundamentally new models, architectures, data representations and algorithms are required.

For these reasons, the NSF sponsored a workshop on hyperbase systems. Workshop participants explored current approaches to hyperbase systems and identified the most important research directions to be pursued. The workshop brought together academic and industrial researchers from the areas of hypermedia, database, information retrieval, and collaborative systems for an intensive two-day workshop. A major goal of the workshop was to foster the cross fertilization of ideas from these diverse and interdisciplinary communities. The workshop was held on October 15 and 16, 1992, in Washington, D.C. This report identifies critical research issues facing hyperbase system researchers, current progress on the issues, potential methods of approach and a general research agenda for the field. The reader is assumed to be familiar with hypermedia systems in general and the various data models and system architectures prevalent in the literature [2, 3, 4, 5, 6, 7, 8, 9, 80, 89].

Sections 2 and 3 give a general overview of hypermedia system architecture, lay out the major dimensions of hypermedia systems and set the context of this report by describing the subspace of these systems discussed at the workshop. Current and future applications of hyperbase technology are illustrated in Section 4, and motivation is given for advanced hypermedia information management systems. Section 5 describes critical issues in five key areas of hyperbase system research, and a general research agenda for the field is presented. Conclusions of the workshop and a look toward the future of this new class of computing systems are provided in Section 6.

## 2. BASIC CONCEPTS AND TERMINOLOGY

Figure 1 provides a general overview of three basic layers of hypermedia system architecture. The application layer contains programs that help accomplish the user's goals. Typical applications are text and graphics editors, mail tools, on-line help systems, case tools, document preparation systems and multimedia presentation tools. The hyperbase layer provides a hypermedia data model to the applications layer and interacts with the storage layer to persistently store the hypermedia data model abstractions, including multimedia data, contexts, nodes, links, anchors, etc. Typical examples of hyperbase systems include HB2 [91], CHS [96], Hyperion [120], Hyperform [115] and DGS [101]. The storage layer is responsible for the persistent storage and retrieval of data. Typical examples of storage managers include distributed file systems and semantic, relational or object-oriented database management systems.

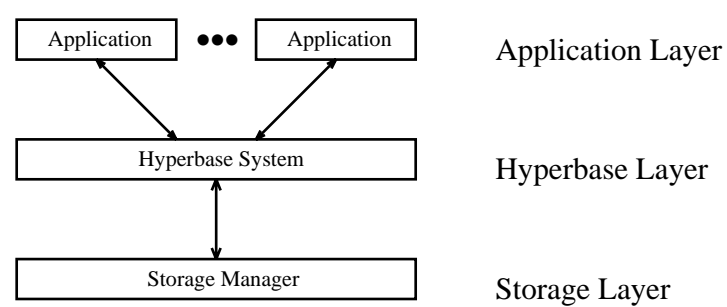


Figure 1. Three basic layers of hypermedia system architecture.

We use the terms *hypertext*, *hypermedia* or *hyperbase* to indicate interconnected information stored by the storage manager on behalf of the hyperbase system. Interconnected information includes multimedia data (objects, nodes, components, composites, etc.) as well as connectivity data (links, anchors, contexts, webs, etc.). The term *hypermedia system architecture* refers to the organization of the three layers (and other layers in any real system) and to the mechanisms involved in communication among the layers. The term *hyperbase system* refers to the software comprising the hyperbase layer of the hypermedia system architecture.

The three layer architecture of Figure 1 is analogous to the classic database system architecture; thus, the term *hyperbase system* is appropriate. It should be noted, however, that the storage manager may not be a database system, and the hyperbase system may act to present a distributed file or graph system model. In this latter case, the term hyperbase system may be somewhat misleading, and a term such as *distributed hypergraph system* may be more appropriate. In this report, we will use the term *hyperbase system* to refer to both cases. Although the hyperbase layer is currently separated from the storage layer, these two layers may be merged in future architectures.

### 3. CONTEXT OF THE WORKSHOP

Hyperbase and hypermedia system architectures vary along many dimensions. In order to characterize the types of hyperbase systems discussed at the workshop, it is necessary to describe the more prominent of these dimensions.

**Scale.** (Hyperbase and hypermedia system architecture.) This dimension refers to the size of the hyperbase, not only in terms of the number of bytes but also in terms of the number of objects (nodes, links, anchors, etc.) and composite objects (versions, contexts, views, etc.) managed. Scale ranges from small sizes of a few megabytes for interactive novels or manual sets, through medium sizes of a few hundred gigabytes for multimedia presentations of digital movies or the page images of a large publisher's journal offerings, to large sizes of a few hundred terabytes for future digital libraries. Scale may also refer to the number of simultaneous users the hypermedia system architecture must support. The number of users may vary from one to tens of thousands.

**Open versus Closed.** (Hypermedia system architecture.) A closed hypermedia system provides a fixed set of encapsulated applications. These applications are normally tightly integrated with the hyperbase system. An open hypermedia system provides a protocol that allows any application to participate in hypermedia services. These applications are loosely integrated with the hyperbase system. Openness ranges from no hypermedia data model restrictions on the applications, to the situation in which the hypermedia data model is embedded in the underlying hyperbase and imposed by the protocol on the applications.

**Centralized versus Distributed.** (Hyperbase system and hypermedia system architecture.) In terms of the hyperbase system, this dimension describes whether the data managed by the hyperbase system resides on one machine's store (centralized) or on multiple machines' stores (distributed). In terms of the hypermedia system architecture, this dimension describes where the application and hypermedia system processes execute - either all on the same machine (centralized) or on multiple machines (distributed).

**Homogeneous versus Heterogeneous.** (Hyperbase system.) This dimension describes whether the same (homogeneous) or different (heterogeneous) hypermedia data models are provided by the hyperbase system(s). In addition, a multi-hyperbase system is composed of more than one hyperbase system and may be homogeneous or heterogeneous.

**Extensibility.** (Hyperbase system.) This dimension describes whether the data model of the hyperbase system can be extended by adding new data abstractions and operations. Dynamic extensibility allows this modification of the data model to occur at run time.

**Computational.** (Hyperbase system and hypermedia system architecture.) This dimension describes whether the architecture supports computation of data model elements such as nodes, links, anchors and virtual structures. The computational capability could be provided by the hypermedia data model abstractions, other layers of the hypermedia system architecture, or by the hyperbase system.

**Method of interaction.** (Hypermedia system architecture.) Single-user hypermedia systems support one user's interaction with the hyperbase system. Multi-user hypermedia systems support multiple users interacting in isolation with the hyperbase system (normal notions of transparency). Collaborative hypermedia systems support multiple users engaged in a shared task and interacting cooperatively (jointly, with awareness) with the hyperbase system.

Given these seven major dimensions, we can characterize the hyperbase system architectures discussed at the workshop as large-scale (tens of terabytes), open, distributed, heterogeneous and extensible with capabilities for computation and collaboration. The workshop concentrated on hyperbase systems and did not include hypermedia system architectures or user-interface issues except as they might impact the hyperbase system.

## 4. APPLICATIONS OF HYPERBASE SYSTEMS

Discussion of typical application environments at the workshop included archival digital libraries, large-scale collaborative systems, medical information systems and large engineering enterprises. Two workshop participants were invited to give presentations on current and future applications of hyperbase technology. Edward Fox presented his work on electronic libraries and Steven Poltrock described how Boeing is currently using hyperbase technology in large engineering enterprises and the future capabilities that hypermedia system architectures must have in order to be useful in these environments. In the remainder of this section, we illustrate the potential of hyperbase technology by summarizing these presentations.

### 4.1 Edward Fox on Electronic Libraries

Two projects were presented in which the LEND (Large External Object-Oriented Network Database) [24, 42] system is being used as the underlying technology for an electronic library. In the first project, LEND is being used in an effort to build a production level system called MARIAN in which 20,000 students will be annotating the card catalog of a million volume library. This is a large application with millions of nodes of card catalog information and millions of links among the nodes. The second project, Project Envision, is a three year, NSF-sponsored effort to explore the whole range of research problems of electronic libraries for a particular discipline and

build (in this case) an electronic library for the area of computer science. This is a large project involving a distributed and heterogeneous collection of machines and their stores and a wide variety of media (tape, CD-ROM, magnetic disk, optical disk, paper, etc.) and formats (video, audio, scanned images, simulations, animations, hypermedia, on-line databases, etc.).

LEND is an object-oriented (C++) hyperbase system specialized for supporting semantic networks, hypertext, hypermedia and information retrieval. LEND contains programming and query language interfaces and has been engineered to manage millions of objects ranging in size from very small to very large. In LEND, objects may be persistent or temporary and primitive or composite. Performance has been a key issue and LEND contains optimal algorithms for minimal perfect hash functions and order preserving minimal perfect hash functions for very large and very small sets. These hash functions allow indexing over large object spaces with rapid search capabilities. LEND maintains the perspective of an underlying graph structure of nodes and links, giving integration from the bottom up. This underlying graph is always available and higher level graphs, composites, paths and views are layered on top of this graph.

Basic architectural principles and the main technical challenges for future electronic libraries were described. In terms of the architecture of electronic libraries, the following six principles were enumerated: 1) we must use declarative representations such as the SGML and HyTime standards, 2) the components we are dealing with should be thought of as objects (e.g., multimedia objects, mathematical expressions, links), 3) we must separate the electronic library from its interfaces (separate the server from its clients) and use standard protocols between the two (e.g., an extended Z39.50 [11]), 4) we should use advanced retrieval algorithms and proven AI techniques (e.g., inference networks), 5) we must have open systems, and 6) we must have task-oriented, user-centered interfaces. In addition to the tough societal problems of intellectual property rights, copyright protection, economic models and social effects, the main technical challenges for future electronic libraries were given as: 1) integration of hypermedia and information retrieval techniques - since the scale of the electronic library requires personalized, intelligent and fast searches (integrating browsing, link and path traversals with search methods), 2) representation, storage and retrieval of multimedia data including the need for media and format conversion and rendering into multiple forms (e.g., to deliver different levels of quality of service for digital video), 3) scaling to millions of users and millions of nodes and links, 4) integration of distributed information servers and name spaces, 5) transparent handling of tertiary storage, and 6) storage and retrieval of an enormous variety of potentially very small and very large complex objects.

## **4.2 Steven Poltrock on Large Engineering Enterprises**

Boeing represents an environment that is already becoming an important market for hypermedia technology and could become a much larger market if the technology would better address the needs of large engineering enterprises [77]. At Boeing, hyperbases are mainly being used in two ways: to deliver information to customers electronically and to deliver information within the company.

The delivery of electronic information to the customer is occurring (or will be in the very near future) in three main ways. First, Boeing's next commercial airplane, the 777, will be delivered with interactive maintenance training manuals. Mechanics will learn how airplane systems work by watching system simulation models while interacting with digital control panels. These simulation models will be linked to the maintenance documentation that describes the systems. Second, on-board digital libraries will contain the operations and maintenance manuals and will

maintain a complete history on all the maintenance that has been done on the airplane. Last, all the maintenance manuals themselves are being developed as on-line deliverable documents that will be made available in different media to different customers.

Hypermedia is also being used internally within Boeing to improve work processes. Boeing maintains a library that defines standard parts and materials used in its products. Instead of designing new parts, engineers can often retrieve part descriptions from this library. Until recently, these descriptions were kept in different sets of bound volumes that were placed in hundreds of locations. This paper-based library was hard to maintain and use. Today a hyperbase system provides engineers with easy on-line access to the standard parts library. In another example, a hypermedia system provides video to the assembly line worker to review how a task is done or to begin learning about a new task.

Current use of hypermedia is limited, however, because it is not well integrated in the everyday working environment of engineers at Boeing. Current hypermedia systems are delivered on a single platform, not the multiple heterogeneous platforms found in the workplace. The information is generally created with some sort of special editor, not the design and analysis tools used by engineers. A small number of authors prepare the materials with no support for multiple authors contributing collaboratively to the same material. The information is intended to be delivered to many readers, who generally have a completely different interface than the authors with no capability for modification or annotation.

Hypermedia can do much more than deliver information. Engineering information is richly interrelated and these relationships could be made explicit using hypermedia. Hypermedia links coupled with powerful navigational tools would help engineers see how their work is interrelated with other aspects of the overall product development and re-use information from previous products. The insights gained from these views would reduce development time and costs and increase product quality. Hypermedia can be used to organize the expanding and evolving body of information created in large engineering projects.

Engineers use highly specialized applications and should be able to establish links within and among objects created by these tools as well as to those objects created with more generic tools. In addition, collaborative work is an essential aspect of engineering today. Support for engineering teamwork will require open hypermedia systems designed to support collaboration.

## **5. KEY AREAS OF HYPERBASE SYSTEMS RESEARCH**

The steering committee asked several workshop participants to present their current research with the objective of exposing convergent and divergent approaches to hyperbase system design. These presentations served to focus and stimulate discussion on the issues. The participants considered the following key areas of hyperbase systems research: models and architectures; node, link, and structure management; integration of information retrieval; version control; and concurrency control, transaction management, and notification control. The following sections contain a summary of the invited presentations and open discussion on the issues in each of the key areas. The research agenda is given as a set of open research issues and comments in each area. Existing and potential approaches to unresolved issues are presented where appropriate.

## 5.1 Models and Architectures

This area is concerned with the hypermedia data model implemented by the hyperbase system and the overall hypermedia system architecture. Due to the importance and fundamental nature of this area, a half-day of the two day workshop was devoted to this topic.

### 5.1.1 Summary of Invited Presentations

Four research laboratories were invited to present their latest research: GMD-IPSI's cooperative hypermedia systems group, Texas A&M's hypermedia research laboratory, University of North Carolina's textlab/collaboratory, and Aalborg University's programming systems laboratory.

**GMD-IPSI.** Helge Schütt described the latest research on GMD-IPSI's Cooperative Hypermedia Server (CHS) [96] which is based on extensions to their earlier HyperBase system [95]. This client-server architecture follows the standard three layer model with the Sybase [108] relational database management system serving as the storage manager. The Sepia collaborative hypermedia authoring environment [107] is the main client application. The data model of HyperBase consists of: *nodes* containing multimedia content, *links* which may be attached to nodes, links, or composites, *composites* which are partially ordered sets of references to other objects (nodes, links, or composites), *attributes* which are name/content pairs attached to any object, and system wide *object identifiers*. In order to support collaboration, the data model of HyperBase was extended with user-controlled locks (called activity markers) and a notification mechanism. CHS also maintains collaboration information such as event subscriptions.

Clients interact with the CHS by retrieving objects through a virtual memory concept; creating, modifying, or deleting objects on the client side; and storing modifications back to the CHS. Clients also send out update notifications to a broadcast server which then broadcasts the event to all clients. Synchronization among clients occurs through the use of *transactions* in the centralized database, *semaphores* in the client processes, *activity markers* on hypermedia objects, and *update notifications* from the broadcast server. Clients are also responsible for defining the following four policies: *caching* - how much data is transferred from the database to the client process; *storage* - which update is stored to the database at what point in time; *update* - when are update requests sent and when are clients willing to receive update requests; and *locking* - when are database transactions needed and how are activity markers interpreted?

**Texas A&M.** John Leggett and John Schnase described the SP2 open hypermedia system architecture [94] and the HB2 hyperbase system [91]. SP2 implements an object-based architecture for distributed, inter-application linking and is an instantiation of a conceptual model of hypermedia that abstracts information, structure, and behavior. SP2 allows hypermedia connections to be forged and "navigated" among applications that are able to participate in a distributed link services protocol. The functionality of the system is realized by client/server relationships among several software components: *Participating Applications*, a *Link Services Manager* (LSM), and the *HB2 hyperbase system* which provides persistent storage for objects and structural data. *Participating applications* must be capable of handling persistent selections (see below) and communicating with the LSM through a standard protocol. Text, graphics and bitmap editors have been developed as participating applications and the Athena Text Widget has also been modified to make Xedit an SP2 participating application [39]. The *link services manager* is a decentralized server process that provides run-time support for interapplication linking. LSM coordinates the interprocess communication required to implement hypermedia functionality. *HB2* is a centralized, multi-user hyperbase system consisting of five subsystems: the hyperbase session

manager, object manager, association set manager, off-line services manager, and the storage manager. HB2 uses the Postgres extended relational database system as its storage manager [91, 93, 104].

The SP2 model of hypermedia consists of six elements: *applications*, *components*, *persistent selections*, *anchors*, *links*, and *associations*. The first three describe the information managed by the system. *Applications* are programs. *Components* are the data or information manipulated by the applications. *Persistent selections* are selections within components that persist between application sessions and can be accessed at a later time. The remaining three components of the model are essential for hypermedia functionality. *Anchors* and *links* are processes in the operating systems sense of the word. They are programs or program units that can be independently scheduled by the underlying operating system in order to accomplish a task. In SP2, they implement the behaviors that characterize hypermedia, such as customized views or traversal. Anchors are associated with persistent selections, and links are associated with anchors, thereby completing connections among persistent selections. The relationships among these elements are called *associations*. Unlike anchors and links, associations are structural entities, collections of identifiers that tie elements together. In this model, the network structure of hypermedia results from the connections forged among persistent selections. An information system based upon this view of hypermedia allows the integration of diverse applications, anchors, and links under a common hypermedia model. The clear evolutionary trajectory of this work is toward the eventual inclusion of hypermedia functionality in the kernel of a network-based, real-time, multimedia operating system.

**UNC.** John Smith and Doug Shackelford described the Artifact-Based Collaboration (ABC) system and the Distributed Graph Storage (DGS) component of the ABC. Researchers at UNC's collaboratory are studying how distributed groups work together to build large, complex structures of ideas and how they merge their ideas and efforts to build a collaborative artifact. The ABC system is being built to support this collaborative environment. ABC has six key components [64, 102]: the distributed graph storage system, a set of graph browsers, a set of application programs, a shared window conferencing facility, real-time video and audio, and a set of protocol tools for studying group behaviors and strategies. DGS provides persistent and sharable storage for the ABC system.

The distributed graph storage system has the following layered architecture: the *Application Layer* contains application-specific programs, the *Application Programming Interface* exports the DGS's graph-oriented data model, the *Graph-Cache Manager* (GCM) implements the data model and performs local caching, and the *Storage Layer* is responsible for permanently storing results. The data model of the DGS consists of *Nodes*, *Links*, and *Subgraphs*. Two mechanisms are provided for storing information within nodes: *attributes* are typed, named variables for storing fine-grained information and node *content* is designed to reference larger amounts of information. Node content can either be a stream of bytes (accessed using a file metaphor) or a composite object called a *subgraph* (accessed using a graph metaphor). A subgraph is defined as a subset of the nodes and links in the artifact that is guaranteed to be consistent with its type (lists, trees, graphs). Two classes of *links* are defined between nodes: structural (used to store the essential structure of an artifact) and hyper-structural (used to represent relationships that cut across the basic structure). Links can have both attributes and content associated with them. The data model also defines that hyperstructural links can be *anchored* to part of a node's content.

The design of DGS is not based on the technology of database systems but rather on distributed file system technology. These researchers believe that a hypermedia storage system might be the next generation distributed file system. The following excerpt [101] gives their rationale:

Given an artifact composed from small elements and user access via interactive browsers, we believe many characteristics and access patterns of objects will strongly resemble those observed in distributed file systems supporting software teams using workstations [12]. Our design is based on the notion that a scalable implementation can be achieved by applying design principles such as local caching, bulk-data transfer, and minimal client-server interactions pioneered in high-performance, scalable file systems like AFS [58], Sprite [82], and Coda [70]. We also model our approaches to data consistency, concurrency semantics, and replication after these distributed file systems. This provides a sufficient level of function to users without requiring the full complexity of mechanisms (e.g., distributed transactions) used in database systems.

*Aalborg.* Uffe Wiil presented the latest research on the Hyperform system [115]. Hyperform can be categorized as an extensible, object-oriented database system that has been specialized to support the rapid prototyping, development and testing of hyperbase systems. Hyperform implements basic hyperbase services that can be tailored to provide specialized hyperbase support. The system is based on an internal computational engine that provides an object-oriented extension language in which new data model objects and operations can be added at run time. These capabilities provide design autonomy to the hyperbase designer by allowing the designer to choose (even at run time) the most appropriate location (client or server side) for hyperbase operations. Hyperform provides a small class hierarchy which can be specialized using multiple inheritance to support many different hyperbase configurations. The built-in classes provide basic services such as concurrency control, notification control (events), access control, version control, and search and query without introducing a particular hypermedia data model or special design policies. Since there are no restrictions on the number of classes and objects, it is possible to have more than one data model and hyperbase configuration running in Hyperform at the same time. This allows heterogeneous hyperbases to coexist concurrently and makes Hyperform the first multi-hyperbase system.

Hyperform is currently implemented as a centralized hyperbase server situated in an extended client-server architecture that supports the development of dynamic, open and distributed systems [114]. This extended client-server architecture includes a tool integrator on the client side. The tool integrator is an extensible, tailorable interface between the Hyperform server and participating hypertext tools, systems and applications. A major advantage of the tool integrator is that it can maintain a shared representation (cache) of important hyperbase data and structure used by the different tools. The extended client-server architecture supports extensibility at two levels in systems: storage and application.

### **5.1.2 Summary of Discussion**

The objective of exposing both common and distinctly different approaches to hyperbase systems was amply achieved in the foregoing invited presentations and the following open discussion. Modeling is somewhat religious by nature and these sessions engendered the most lively debate at the workshop, with participants questioning even the most fundamental notions of hypermedia systems. Several major issues surfaced in the open discussion on models and architectures.

Although the following discussion is separated into several categories, it should be noted that most of these issues are highly interrelated.

***Data model versus process model.*** The fundamental question of whether hypermedia should be modeled solely as a data model or as a combination of a data and a process model was debated. Both views see hyperbases as an intermediate layer between the application layer and the storage layer. Proponents of the data model view consider process to be in the application layer and outside of the model. Proponents of the combined data and process model view require that process abstractions as well as data abstractions be provided from the hyperbase layer. It was noted that most of the hypermedia models in the literature are data models and do not sufficiently specify process. This is an open research area. Hypermedia is as much a user-interface or operating system technology as a database or information storage and retrieval technology. We do not have a formal definition for hyperbases.

***Hyperbase versus distributed graph server.*** Much of the discussion in this session centered around the fundamental question of whether these hypermedia storage systems should be based on the database paradigm or a distributed file system paradigm. The only consensus the group reached was that it was too early to decide this question. It was felt that each research system was exploring a different design space and the various systems should be built and tested to see which design decisions worked and which did not. In the future, we can judge these systems based upon the seven dimensions given earlier. For example, on the scale dimension we can measure: how large a body of material can be handled by the systems, how many simultaneous users, how widely distributed, and with what response time. It may also be true that the intended class of application will determine whether a database or distributed file system paradigm is most appropriate. For example, for large digital libraries a database paradigm may be more appropriate, while for collaborative systems a distributed file system paradigm may be more appropriate. Experimental research is critically needed in this area.

***Heterogeneity.*** At least three types of heterogeneity must be supported. First, we must support the linking of heterogeneous applications. This inter-application linking is at the very heart of open hypermedia systems architectures. Several research groups have defined and implemented linking protocols and anchor models that will support this capability. Research is needed to evaluate the existing anchoring models in order to come to some consensus on appropriate solutions. Second, we must be able to interconnect heterogeneous hardware/operating system platforms. Third, we must be able to link major, separate storage systems. These storage systems could be heterogeneous hyperbase systems, relational database systems, full-text database systems, etc. It is especially important to be able to include linking into existing databases (legacy systems). This is currently an open area of research. Our approaches to this problem have mainly been based upon building hypermedia interfaces to existing database systems. A possible approach would be to review the heterogeneous database systems research on global schemas and ask whether a universal set of hypermedia functions could be built that could be accessed in a uniform manner across storage systems.

**Open research questions.** Several questions were raised but not discussed in any depth. These questions are listed here as open reserach questions.

- Should a hyperbase system provide a hypermedia description language, hypermedia maintenance language, and hypermedia query language analagous to database systems?
- What new file organizations and data structures are required to provide support for the data manipulation and process operations of hypermedia applications (such as complex structural queries or computing transitive closure)? Which storage technology better supports these needs – database or distributed file systems?
- Where is the media in hypermedia? How do we provide persistent anchoring in the various media, not only for presentation but to support the arbitrary linking provided by contexts? How will we guarantee performance across heterogeneous platforms?
- What about standards and protocols? Are we building the next set of legacy systems because we are not adhering to the emerging standards? How does the HyTime [83] standard and the Z39.50 protocol [11] fit into hypermedia system architectures? Are these sufficient or deficient for our purposes?
- Where should the computation go? In the hyperbase, application, or other intermediate layers? What about virtual structures? Should nodes, links, and anchors all be computable objects? How do we specify process in the hyperstructure? How do we give semantics to the hyperstructure?
- Are we ready to try to standardize on the anchoring model for open hypermedia systems? Which software component is responsible for generating the anchor ID? Should there be one or multiple anchoring data structures per object?

## **5.2 Node, Link and Structure Management**

This area is concerned with simple and composite object identity, name space and partitioning services, the efficient management of various data types and link and anchor behaviors, and the maintenance of constraints on the objects and structure of hypermedia to ensure the overall integrity of the network. The workshop chose this latter topic as its focus for discussion.

### **5.2.1 Summary of Invited Presentation**

David Stotts described work done in collaboration with Richard Furuta on the integration of process specifications in hyperstructures. The premise of this research is that the primary distinguishing difference between electronic documents and hyperdocuments is in the process of interaction and that interaction should be specified along with the hyperstructure, thus giving semantics to the raw data structures. This view assigns two fundamental behaviors to different levels of the model: process behavior belongs at the hyperstructure level and rendering behavior (e.g., visual or auditory) belongs at the application level. This model inherently represents the browsing behavior in the document itself, obviating the need for the authoring environment in order to view the hyperdocument as the author intended.

This model views a hypermedia document as a process description, not a data structure. In the Trellis model [45, 105, 106] a Petri net is used as the process description of a hyperdocument. Node contents and buttons are associated with places in the Petri net. Browsing occurs as the tokens move along the transitions. The Trellis model allows for multi-headed and multi-tailed links, multiple processes, concurrency, access control and synchronization.

As a process description, one can view the hyperdocument as a program expression and use techniques from program verification to verify properties or constraints about the hyperstructure of the hyperdocument. One technique for doing this is called model checking [26, 27]. Model checking represents the link structure as a finite state automaton. The execution of this automaton defines a tree of possible event traces. The model checking approach allows one to write specifications about the patterns these traces should adhere to and then quickly tell whether the automaton, that is the document, exhibits the behavior it should.

The claim is that this is a particularly interesting technique to specify and verify dynamic constraints and that the desired behavior is found when you interact with a link structure. It is felt that this technique will at least scale to medium-sized hyperdocuments and that hyperbases are going to have to have the ability to provide a succinct way of specifying this kind of semantics.

### **5.2.2 Summary of Discussion**

The fundamental question of whether a hypertext is really a program was debated with no clear consensus being reached. Discussion also centered around the various ways one might include computation in hypermedia: in the browser or application; in the anchors, links and nodes themselves; or associated with the hyperstructure. It was pointed out that the necessity for virtual structures makes integrity checking difficult. Nevertheless, the ability to check both static and dynamic integrity constraints is a very valuable feature for hyperbase systems. A particularly open area of research is the definition of languages for specifying these integrity constraints and the process of hypertext in general.

## **5.3 Integration of Information Retrieval**

This area is concerned with the integration of information retrieval and hyperbase technologies. This includes various indexing techniques such as multi-level store, distributed, and task-based indexing, the integration of browsing with traditional index-based searching, the use of agency (mediators, guides, critics, constraints), the use of path histories to guide further search, and the specification of hyperbase query languages.

### **5.3.1 Summary of Invited Presentations**

Mark Frisse, Bruce Croft and Tim Oren were invited to present their views on the issues of integrating information retrieval and hyperbase technologies. Unfortunately, Tim became ill and was unable to present his views on integrating agency in hyperbases.

*Mark Frisse on data and people issues.* Mark decomposed his talk into issues that matter for the data in hyperbases and issues that matter for the way people will use the data in hyperbases. He based these issues on his experiences with providing online medical information to practicing

doctors and medical students. Issues for data in hyperbases were covered first. The first issue is whether the data is structured and, if so, how deeply. Mark discovered from his earlier work on the Dynamic Medical Handbook project [43, 44], that many of the browsing capabilities did not make much difference because the structure of the corpus was very broad and shallow. The second issue is whether or not the data has a temporal dimension. If a temporal dimension exists then a method must be devised for representing time and opportunities arise for temporal queries and time-based versioning. The third issue concerns the directionality of links. Bidirectional links provide the opportunity to build indexes such as the science citation index. The fourth issue is whether or not the declaration of knowledge is implicit or explicit. If the knowledge is explicit then one stands a better chance of reasoning over the information. The fifth issue is whether the data is indexed and the vocabulary controlled. Mark believes it is just a matter of time until controlled indexing is going to disappear in the field of medicine. The last issue is whether the system learns from your interactions or is passive. The question here is whether you get enough training instances to make that sort of feedback applicable for the individual.

Mark then described issues that matter for the way people will use the data in hyperbases. The first issue is whether the information is being used by an individual or a group. This is especially important in systems that learn from your interactions. Do you have a single person training the network or a group? The second issue is the awareness level of the people. Mark has found that (in experiences with twenty or so online medical books) if the people know the structure of the book they do not need anything else and if they do not know the structure of the book, they do not have a clue of where to go and that is what absolutely necessitates some sort of outside global searching mechanism. The third issue is process. Are people aware that there is an order to the information? The fourth issue is semantics. Are people aware that different terms mean different things to different people? The fifth issue is one of deepness of knowledge. You can certainly use the same corpus of knowledge, but the novice will require a very different explanation than the expert. The sixth issue is domain. Is the item of momentary importance or is it something that is an area of traditional scholarship? The last issue is the issue of copyright.

Finally, Mark described the need for electronic libraries in medicine. One need is driven by the skyrocketing cost of paper journals, and another by the need to support alternate means of information presentation for things such as the human genome and three dimensional color graphic simulations in dynamic documents.

***Bruce Croft on integration issues.*** Bruce pointed out that many of the issues in integrating information retrieval and hyperbase technologies are shared with the previous attempts to integrate information retrieval and database systems. In the "old days" attempts were made to implement an information retrieval system on top of a database system. The motivation for this is quite simply that we need systems that can deal with both the structured data of database systems and the unstructured data of information retrieval systems. This need has become more and more important as time has gone by. But there are several problems with this approach. First, databases deal with certainty, while information retrieval systems deal with probability and uncertainty. Second, information retrieval query languages are more complicated than database query languages since they have to deal with issues such as relative importance, related concepts, and feedback. Third, building efficient indexes over very large volumes of text requires very different file organizations than databases provide.

What approaches could we try for integrating information retrieval and hyperbase technologies? In the best of all worlds we could try to generate a unified model. This seems to be a nice goal but rather impractical in the short term due to the installed base of database and information retrieval systems and the fact that this is a very difficult research issue. A second strategy would be to build information retrieval systems as applications on top of hyperbase systems. This approach would suffer from the same concerns (mentioned above) of building information retrieval systems on top of database systems. A third approach would be to integrate information retrieval and hypertext. This seems to be the most promising approach. We could integrate tools that would build structure automatically from text by using similarity comparison techniques and make these tools available to the hypertext application programmer. We could integrate the use of links in information retrieval searches. Information retrieval usually assumes that the collection is a set of unrelated objects; this is not true in hypertext. How do the links impact the information retrieval searching capabilities? We could also integrate the browsing function of hypertext with the searching function of information retrieval and use these in an alternating fashion. How would this integration be accomplished at the user interface?

Finally, Bruce presented the following challenges. In the area of models, we need models that can express how an object inherits meaning from all the different objects that are associated with it. In the area of queries, composite objects can play a key role in the retrieval of related objects. For example, we should be able to retrieve objects based on their content or their sub-objects content and we should be able to combine type specifications as well as content specifications in queries. We should also have associated hypermedia query languages for the hypermedia models.

### **5.3.2 Summary of Discussion**

The open discussion for this session covered a very broad range of highly interrelated topics.

***Hyperbase query languages.*** The need for hyperbase query languages surfaced many times throughout the discussion. This is a particularly open area of research. Hyperbase query languages are going to be challenging. We want to pose queries such as, "I want objects which are pointing at three other types of objects and whose content is about something." This will require a structural query combined with a content query, preferably in the same language. We will need something like a probabilistic object-oriented algebra to really solve the problem properly. The GraphLog project at Toronto [31] had a very nice solution to some of the structural queries. But a tough question that remains is how to integrate the ranking by probabilities that information retrieval algorithms give with the rest of the query.

***Hypertext links and multiple media.*** The types of links people construct when authoring hypertext are qualitatively different than the sort of information one can extract with a content-based index. We would like to have the ability to have both content searches and searches based on these higher level associations in our systems. Some participants noted that most of information retrieval is primarily based upon text and that new media offer opportunities to look at alternative indexing mechanisms. Other participants noted that many times, even with textual databases, it is not the content that is important but the text that explains the content.

***What could we do in the short term?*** There are a few things that could be done in the short term to make an integrated information retrieval/hyperbase system. The simplest thing to do is restrict the query language to selection on object type, then build the appropriate tools that know how to do

indexing of objects by their content. These tools have to have some way of knowing when it is time to index. The major concern would be the associated object problem - the general idea that objects can be retrieved according to the meaning of related objects. This could be a pitfall depending on how sophisticated we want to make the system. But we could easily do things such as indexing time, figures, and pictures by text nodes that are connected to them. We would just be using the composite object structure and/or the links to indicate objects that represent the contents of the objects of interest.

**Open research questions.** Several questions were raised but not discussed in any depth. These questions are listed here as open research questions.

- How can we guide information access by prioritizing link choices? By using path histories, personal interest profiles, points-of-view, community-based indexing, task-based indexing? This becomes increasingly important as the number of choices increases.
- How can we handle non-monolithic, distributed indexes over distributed information spaces? Especially when the information is very dynamic. Inverted indexing doesn't seem to be the answer. Can semantic indexing help?
- How can information retrieval help during the authoring aspects of hypertext?
- What kinds of file structures are appropriate for structural queries posed against external link bases?

## **5.4 Version Control**

This area is concerned with the version control mechanisms that are provided in the hypermedia systems architecture. This includes version control data models, versioning hypermedia data and structure, version creation and selection, temporal versus functional versioning, immutability of versions, merging of versions, and partitioning the versioning among the layers of the architecture.

### **5.4.1 Summary of Invited Presentations**

Three research laboratories were invited to present their latest research: Texas A&M's Hypermedia Research Laboratory, GMD-IPSI's cooperative hypermedia systems group, and Aalborg University's Programming Systems Laboratory.

**Texas A&M.** David Hicks presented an overview of version control in open hypermedia systems [55, 56]. He noted that most of the appropriate prior research comes from the fields of software configuration management and computer-aided design and that most of the previous version control systems are highly tuned for a particular application domain. In open hypermedia system architectures there are two main loci of versioning requirements: the application layer which represents a wide diversity of versioning needs and the hyperbase layer which should provide an application-independent, policy-neutral versioning mechanism. When partitioning the versioning functionality between these two levels, one must consider each facet of version control functionality carefully to determine where it can best be implemented.

Dave proposed several broad categories of functionality that should be implemented at the hyperbase level. The first category is an appropriate data model. In a version control system, the data model determines how objects are allowed to evolve. The version control data model at the hyperbase level should be capable of supporting the physical object evolution process. The tree data model is a good place to start an investigation of appropriate data models [74, 110]. The second category concerns storage considerations. The introduction of versioning requires the management of a large number of objects. We need efficient storage techniques and compression algorithms. As new client applications are developed, we need to consider the effects that new data types and media have on the storage mechanisms and develop algorithms that help us control the amount of storage needed. The third category is internode referencing. When versioning is in effect, object identities must discriminate particular versions of objects. The final category is a composition mechanism. A composition mechanism is important for versioning for at least two reasons: to group collections of objects which are versions of the same object and to assist the application developer in coping with multiple versions of the same object.

Several categories of version control functionality should be implemented at the application level. The first category at this level (as above) concerns an appropriate data model. The data model at this level represents the need to support object evolution as it is perceived by the end user. This model is likely to be expressed in application-specific abstract terms and may bear little resemblance to the physical level model. The second category is intranode specification. Versioning could result in some of the intranode specifications becoming corrupt. The actions that result in new versions being created could destroy locations that correspond to the endpoint of links. The third category is user interface considerations. Issues in this category are especially application dependent and must be considered carefully because they have a direct impact on the usability of the version control service. The fourth category concerns the proliferation of versions. Versions should only be created when something of importance has changed and the previous state might need to be retrieved. The fifth category is maintenance of valid configurations including baselines. Some systems have objects which are basically composed of other objects, such as a configuration object in a software configuration management system. It is possible for versioning to cause these types of objects to become corrupt. When you create a new version, it might not be correct for that new version to be automatically included into the composing object (configuration). The last category concerns the immutability of versions. This refers to the decision that an application developer must make about the appropriate time to freeze an object. In some application areas, it might be appropriate that once a version is created it is completely frozen and can never be updated over time. There are also degrees of immutability. For example, you might, in some cases, need to be able to modify the contents of an object or alternatively to modify the attributes of an object even after it is frozen.

***GMD-IPSI.*** General problems in providing versioning support for hypermedia applications arise due to the fine-grained, heavily interlinked structure of hyperdocuments. Cognitive overhead problems arise during version creation. Which updates lead to a new version? Which changes should be propagated to cause new versions of referencing objects (implies automatic version creation)? How do you describe versions for later identification purposes? Disorientation problems arise during version selection. Automatic versioning must be understood by the users.

Anja Haake described the Contextual Version Server called CoVer [51] designed to deal with these problems. CoVer provides version control services for application clients of the Cooperative Hypermedia Server (CHS) [96] discussed earlier. CoVer is an application independent version

server allowing both object-centered and task-centered access to versions. Applications define their application-specific datatypes (subclasses of the CHS nodes, links, and composites) in the application interface to CHS and may implement many different versioning policies.

CoVer's basic version management consists of single-state objects (snobs), multi-state objects (mobs), versions, derivation histories, and references to versioned objects. Snobs represent non-versioned objects. Mobs represent versioned objects and contain application-defined state-independent attributes. Mobs consist of a set of versions. Versions represent a state of a versioned object and contain application-defined state-dependent attributes. Versions may be updateable or frozen. Derivation histories record the reuse of content across document boundaries through the *derive new version/object* operations. Referencing versioned objects is a two-step process. The object identifier is given as a static value and the version identifier is given as an arbitrary query. The query may return multiple versions which are considered alternatives with respect to the equality characteristic given by the query.

CoVer's context-based version management consists of tasks and annotations. Tasks maintain the versions of the various objects created and used in the context of performing some job. Eventually tasks maintain the state of a hyperdocument that fulfills the requirements of the respective job; at this point it is natural to freeze the task. Tasks may be composed into task hierarchies and are used to guide automatic version creation, support version identification and partition the information space. Tasks are implemented on top of CHS as a composite object and applications can specify task-related attributes that may be queried and special links representing task relationships that may be navigated. Annotations are used to comment on the content of other objects and many times result in the generation of tasks and versions that react to the comment.

**Aalborg.** Kasper Østerbye described the version control data model of the HyperPro [122] system and presented his views on some of the outstanding issues of version control in hyperbases [121]. Versioning in HyperPro is centered around the notions of version groups and contexts. Version groups collect the individual versions of an object in a graph data model and contexts provide the same functionality as configurations in software configuration management.

Kasper discussed three main performance issues that are open: element selection, version creation, and static versus generic links. These are performance issues in the sense that you want the hyperbase system to be able to handle these issues without incurring additional network traffic. Element selection is concerned with selecting the correct version of an object. If a query-based mechanism is used to select the particular version of an object, it may return several alternatives. We need mechanisms the hyperbase system can use to help disambiguate the query. The use of default contexts can help in this situation. The issue of version creation is tied to the issue of promotion (propagation) of versions to referencing objects. We need mechanisms that the hyperbase system can use to decide when and when not to promote versions. Finally, when objects get versioned, how do you determine which of the links that pointed to the original object should remain as static links and which should be promoted to generic links that point at the version group as a whole? Again, we need mechanisms that will allow this processing to be done on the hyperbase server side. Each of these issues is difficult to address because the policy should be determined on the application (client) side, but the mechanisms should be executed on the hyperbase (server) side.

### **5.4.2 Summary of Discussion**

The discussion in this session centered mainly around the problems of making version control systems that are as efficient as file systems in terms of copy operations and user perceptions. Discussion is summarized as open questions/comments.

#### *Open questions/comments.*

- We need to be able to copy contexts as easily as we can copy files today, but with hypermedia we have many more objects (nodes, links, anchors, composites) to consider. For example, copying a context to make a baseline requires the promotion of generic links to static links. These operations result in an optimization problem that must be addressed at the hyperbase level.
- In some hypermedia systems, it makes as much sense to copy new node versions as to use a delta compression technique because the node granularity is so small that the overhead for keeping a separate delta object doesn't make sense.
- What are the degrees of immutability that make sense at the hyperbase level? Application level?
- Assuming you have composites that reference the exact version of an object with a query, where is the query stored? In the composite? In the surrounding context? In the reference (link?) itself?
- The issue of when to index is obviously intimately connected to version control. In version control you're trying to restrict when you make new versions. Similarly, we want to restrict how often to do new indexing. It's obviously going to be part and parcel of the same problem.
- Versioning must be transparent for legacy systems. What does it take for a legacy system to become version aware?
- How can we simplify the user's view of versioning?
- This is a topic that cries out for a formal semantic description.

## **5.5 Concurrency Control, Transaction Management and Notification Control**

This area is concerned with managing short, long, and very long hyperbase transactions; locking mechanisms and granularity of locking; integrity, sharing, and recovery of data; and event subscription and management.

### **5.5.1 Summary of Invited Presentation**

Traditional concurrency control techniques for database systems (transaction management based on locking protocols) have been successful in many multiuser settings, but these techniques are inadequate in open, extensible and distributed hypertext systems supporting multiple collaborating users [116]. Uffe Wiil described the built-in services of the concurrency control and notification control objects in the Hyperform system [115]. These objects support the use of Hyperform in a

collaborative system setting. The concurrency control object supports the integrity and sharing of data by providing attribute-level and object-level locking. Read access is allowed to locked attributes and objects to support the browsing of networks in shared hypermedia. The concurrency control object also supports transaction management by providing a simple two-phase commit protocol that requires explicit locking of resources and allowing the grouping of Hyperform methods into atomic operations at execution time (data model extensibility). This architecture raises an interesting and open question: to what extent does a centralized client-server architecture with an extensible hyperbase system obviate the need for transaction management?

Notification control allows users to be aware of important actions on the shared network of objects (nodes, links, composites) and supports user-controlled locking and social protocols inherent in collaborative work. The notification control object in Hyperform provides support for collaborative work through event subscriptions and notifications. Event subscriptions can be any Scheme expression involving the objects, attributes, and operations in the Hyperform server.

### **5.5.2 Summary of Discussion**

Again, this is an area that is particularly open for further research. The open discussion in this area turned into a debate on the merits of centralized versus distributed architectures and how one could efficiently implement notification control in these environments. Discussion is summarized as open questions/comments.

#### *Open questions/comments.*

- How will the evaluation of event subscriptions affect performance? What can we do about it?
- Is there any natural partitioning of the event space that the hyperbase can take advantage of?
- Should the hyperbase make distribution (of data, of control) transparent to the user?
- What affect does event granularity have on performance in centralized/distributed architectures?
- How do access and notification control affect each other?

## **6. CONCLUSIONS**

As one can see, the workshop raised more questions than it answered. This is to be expected in a new field. The real challenge now is to address these issues in a disciplined way and to develop consensus as future research clarifies the issues. The participants at the workshop came from many different disciplines and conceptualized hypermedia storage systems in many different ways. At this early stage, we feel that it would be a mistake to arbitrarily choose one point of view over the others. Instead, we should explore those portions of the design space we have each taken and build real systems for real people. We should evaluate the efficiency and user acceptance of the systems to see if these tools make a difference. In the long term, we can synthesize our experiences to see which techniques and mechanisms work best.

A major goal of the workshop was to foster the cross fertilization of ideas from diverse, interdisciplinary communities. We feel that this goal has been achieved. It was apparent that several of the representative research projects were aimed at providing the kinds of services Steven Poltrock had suggested hypermedia system architectures must have in order to be useful in large engineering enterprises. Support for large digital libraries also provides a challenging environment for improving models, scaling-up implementations, and integrating applications and systems. Participants discovered common research interests and a substantial overlap in issues and technical requirements for their systems.

Hyperbase research is obviously at a very early stage. The greatest opportunity to influence the field will occur in the next few years. We should encourage groups to collaborate, share perspectives, and build on each other's work. A coordinated, multidisciplinary research program is needed to define fundamental principles of hyperbase systems. A program of software research is needed to develop better tools for the design and implementation of future hyperbase system architectures. The benefits from a cooperative and coordinated approach would be significant.

## **ACKNOWLEDGEMENTS**

We would like to acknowledge the efforts of Cindy Kunz in the organization and recording of the workshop. The transcript she produced from the tapes was invaluable in preparing this report. This report represents the combined inputs of all participants. We would also like to acknowledge the financial support provided by CRSS Architects, Inc.

## **BIBLIOGRAPHY**

1. Abrial, J. R. Data semantics. In *Data Base Management*, J. W. Klimbie and K. L. Koffemen, (Eds.), North-Holland, Amsterdam, (1974), pp. 1-59.
2. ACM. *Proceedings of the Hypertext '87 Conference*, (Chapel Hill, NC), (Nov. 1987).
3. ACM. Special Issue on Hypertext. *Comm. ACM* 31, 7 (July 1988).
4. ACM. Special Issue on Hypertext. *ACM Trans. Off. Inf. Syst.* 7, 1 (Jan. 1989).
5. ACM. *Proceedings of the Hypertext '89 Conference*, (Pittsburgh, PA), (Nov. 1989).
6. ACM. *Proceedings of the Third ACM Conference on Hypertext (Hypertext '91)*, (San Antonio, TX), (Dec. 1991).
7. ACM. Special Issue on Hypermedia. *Comm. ACM* 35, 1 (Jan. 1992).
8. ACM. *Proceedings of the Fourth ACM Conference on Hypertext (ECHAT '92)*, (Milan, Italy), (Nov. 30-Dec. 4, 1992).
9. ACM. *Proceedings of the ACM Conference on Hypertext (Hypertext '93)*, (Seattle, WA), (Nov. 1993), (forthcoming).

10. Afrati, F., and Koutras, C. A hypertext model supporting query mechanisms. In *Hypertext: Concepts, Systems, and Applications. Proceedings of the European Conference on Hypertext*, (France), A. Rizk, N. Streitz, and J. Andre, (Eds.), Cambridge University Press, Cambridge, U.K., (Nov. 1990), pp. 52–66.
11. American National Standards Institute. Information Retrieval Service and Protocol. American National Standard Z39.50-1988, Transaction Publishers, New Brunswick, NJ, (Jan. 1988).
12. Baker, M. G., Hartman, J. H., Kupfer, M. D., Shirriff, K. W., and Ousterhout, J. K. Measurements of a distributed file system. *Operating Systems Review, Special Issue: Proceedings of the 13th ACM Symposium on Operating Systems Principles*, (Pacific Grove, CA), 25, 5, (1991), pp. 198-212.
13. Batory, D., and Kim, W. Modeling concepts for VLSI CAD objects. *ACM Trans. Database Syst.* 10, 3 (Sept. 1985), 322–346.
14. Batory, D. S., Barnett, J. R., Garza, F. F., Smith, K. P., Tsukauda, K., Twichell, B. D., and Wise, T.E. GENESIS: A reconfigurable database management system. *IEEE Trans. Softw. Eng.* SE-12, 11 (Nov. 1986), 1711-1730.
15. Beeri, C., and Kornatzky, Y. A logical query language for hypertext systems. In *Hypertext: Concepts, Systems, and Applications. Proceedings of the European Conference on Hypertext*, (France), A. Rizk, N. Streitz, and J. Andre, (Eds.), Cambridge University Press, Cambridge, U.K., (Nov. 1990), pp. 67-80.
16. Berre, A. J. SOOM and Tornado<sup>\*</sup>: Experience with database support for object-oriented applications. In *Advances in Object-Oriented Database Systems: Proceedings of the 2nd International Conference on Object-Oriented Database Systems*, K. Dittrich, (Ed.), Springer-Verlag, New York, (Sept. 1988), pp. 104-109.
17. Berre, A. J., and Anderson, T. L. The hypermodel benchmark. In *Object-Oriented Databases with Applications to CASE, Networks, and VLSI CAD*, R. Gupta and E. Horowitz, (Eds.), Prentice-Hall, Englewood Cliffs, NJ, (1991), 75–91.
18. Bracchi, G., Paolini, P., and Pelagatti, G. Binary logical associations in data modelling. In *Modelling in Data Base Management Systems*, North Holland, Amsterdam, (1976), pp. 125-148.
19. Brodie, M. L., and Ceri, S. On intelligent and cooperative information systems: A workshop summary. *International Journal of Intelligent and Cooperative Information Systems* 1, 3, (Fall 1992).
20. Bruza, P. D. Hyperindices: A novel aid for searching in hypermedia. In *Hypertext: Concepts, Systems, and Applications. Proceedings of the European Conference on Hypertext*, (France), A. Rizk, N. Streitz, and J. Andre, (Eds.), Cambridge University Press, Cambridge, U.K., (Nov. 1990), pp. 109-122.
21. Campbell, B., and Goodman, J. HAM: A general-purpose hypertext abstract machine. *Commun. ACM* 31, 7 (July 1988), 856-861.
22. Caplinger, M. An information system based on distributed objects. *Proceedings of the OOPSLA '87 Conference*, (October 1987), pp. 126-137.

23. Carey, M., DeWitt, D., Richardson, J., and Shekita, E. Object and file management in the EXODUS extensible database system. *Proceedings of the Twelfth International Conference on Very Large Data Bases*, (Kyoto, Japan), (August 1986), pp. 91-100.
24. Chen, Q. F. An object-oriented database system for efficient information retrieval applications. Ph.D. Dissertation, Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA, (March 1992).
25. Christodoulakis, S., Theodoridou, M., Ho, F., Papa, M., and Pathria, A. Multimedia document presentation, information extraction, and document formation in MINOS: A model and a system. *ACM Tran. Off. Inf. Syst.* 4, 4 (October 1986), 345-383.
26. Clarke, E. M., Emerson, E. A., and Sistla, A. P. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. on Programming Languages and Systems*, 8, (1986), 244-263.
27. Clarke, E. M., and Grumberg. Research on automatic verification of finite-state concurrent systems. *Ann. Rev. Comput. Sci*, 2, (1987), 269-290.
28. Clifton, C., and Garcia-Molina, H. Indexing in a hypertext database. *Proceedings of the 16th Conference on Very Large Data Bases*, (Brisbane, Australia), (August 1990), pp. 36-49.
29. Clitherow, P., Riecken, D., and Muller, M. VISAR: A system for inference and navigation of hypertext. *Proceedings of the Hypertext '89 Conference*, (Pittsburgh, PA), (Nov. 1989), pp. 293-304.
30. Conklin, J. Hypertext: An introduction and survey. *Computer* 20, 9 (Sept. 1987), 17-41.
31. Consens, M. P., and Mendelzon, A. O. Expressing structural hypertext queries in GraphLog. *Proceedings of the Hypertext '89 Conference*, (Pittsburgh, PA), (Nov. 1989), pp. 269-292.
32. Croft, W. B., and Turtle, H. A retrieval model incorporating hypertext links. *Proceedings of the Hypertext '89 Conference*, (Pittsburgh, PA), (Nov. 1989), pp. 213-224.
33. Crouch, D. B., Crouch, C. J., and Andreas, G. The use of cluster hierarchies in hypertext information retrieval. *Proceedings of the Hypertext '89 Conference*, (Pittsburgh, PA), (Nov. 1989), pp. 225-237.
34. Deheneffee, C., Hennebert, H., and Paulus, W. Relational model for a database. *Proceedings of the IFIP Congress*, (1974), pp. 1022-1025.
35. Delisle, N., and Schwartz, M. Neptune: A hypertext system for CAD applications. *Proceedings of the ACM International Conference on the Management of Data (SIGMOD)*, (1986), pp. 132-143.
36. DeWitt, D. J., Fattersack, P., Maier, D., and Velez, F. A study of three alternative workstation-server architectures for object oriented database systems. *Proceedings of the 16th International Conference on Very Large Data Bases*, (Brisbane, Australia), (August 1990), pp. 107-121.
37. Dittrich, K., and Dayal, U. (Eds.). *Proceedings of the International Workshop on Object-Oriented Database Systems*, (Pacific Grove, CA), (Sept. 1986).

38. Dittrich, K. R., Gotthard, W., and Lockemann, P. C. DAMOKLES – A database system for software engineering environments. *Proceedings of the IFIP Workshop on Advanced Programming Environments*, (Trondheim, Norway), R. Conradi, R. M. Didriksen, and D. H. Wanvik, (Eds.), Springer-Verlag, New York, (June 1986), pp. 353-371.
39. Drufke, J. E., Leggett, J. J., Hicks, D. L., and Schnase, J. L. The derivation of a hypertext widget class from the Athena text widget. Department of Computer Science Technical Report No. TAMU 91-002, Texas A&M University, College Station, TX, (1991).
40. Fishman, D., Beech, D., Cate, H., Chow, E., Connors, T., Davis, J., Derrett, N., Hoch, C., Kent, W., Lyngbaek, P., Mahbod, B., Neimat, M., Ryan, T., and Shan, M. IRIS: An object-oriented database management system. *ACM Trans. Off. Inf. Syst.*, 5, 1 (January 1987), 48-69.
41. Fox, E. A. Advances in interactive digital multimedia systems. *IEEE Comput.* 24, 10, (1991), 9–21.
42. Fox, E. A., France, R. K., Sahle, E., Daoud, A., and Cline, B. E. Development of a modern OPAC: From REVTOC to MARIAN. *Proceedings of SIGIR '93, 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, (Pittsburgh, PA), (June 27-July 1, 1993), pp. 248-259.
43. Frisse, M. Searching for information in a hypertext medical handbook. *Commun. ACM* 31, 7 (July 1988), 880-886.
44. Frisse, M. E., and Cousins, S. B. Information retrieval from hypertext: Update on the dynamic medical handbook project. *Proceedings of the Hypertext '89 Conference*, (Pittsburgh, PA), (Nov. 1989), pp. 199-212.
45. Furuta, R., and Stotts, P. D. The trellis hypertext reference model. *Proceedings of the NIST Hypertext Standardization Workshop*, National Institute of Standards and Technology, (Gaithersburg, MD), (Jan. 1990), pp. 83-93.
46. Gallagher, L., Furuta, R., and Stotts, P. David. Increasing the power of hypertext search with relational queries. *Hypermedia* 2, 1 (1990), 1-14.
47. Goldfarb, C. F., Ed. International Standard 10744, Information Technology- Hypermedia/Time-based Structuring Language (HyTime), International Standards Organization, (April 1991).
48. Griffith, R. L. 1982. Three principles of representation for semantic networks. *ACM Trans. Database Syst.* 7, 3 (Sept. 1982), 417-442.
49. Griffith, R. L. 1989. Semantic-network databases for VLSI design. *Proceedings of the Seventh British National Conference on Databases (BNCOD 7)*, (Heriot-Watt University), M. H. Williams, (Ed.), Cambridge University Press, Cambridge, U.K., (July 1989), pp. 5-36.
50. Gupta, R., and Horowitz, E. *Object-Oriented Databases with Applications to CASE, Networks, and VLSI CAD*. Prentice-Hall, Englewood Cliffs, NJ, (1991).
51. Haake, A. CoVer: A contextual version server for hypertext applications. *Proceedings of the European Conference on Hypertext (ECHT '92)*, (Milan, Italy), (Nov. 1992), pp. 43-52.

52. Hainaut, J. L., and Lecharlier, B. An extensible semantic model of database and its data language. *Proceedings of the IFIP Congress*, (1974), pp. 1026-1030.
53. Halasz, F. Reflections on NoteCards: Seven issues for the next generation of hypermedia systems. *Commun. ACM* 31, 7 (July 1988), 836-852.
54. Halasz, F., and Schwartz, M. The Dexter hypertext reference model. *Proceedings of the NIST Hypertext Standardization Workshop*, National Institute of Standards and Technology, (Gaithersburg, MD), (Jan. 1990), pp. 95-133.
55. Hicks, D. L., Leggett, J. J., and Schnase, J. L. Version control in hypermedia: An open systems perspective. Department of Computer Science Technical Report No. TAMU-HRL 93-001, Texas A&M University, College Station, TX, (1993).
56. Hicks, D. L. VOM: A version control model for open, extensible hypermedia architectures. Dissertation, Department of Computer Science, Texas A&M University, College Station, TX, (1993).
57. Horowitz, E., and Wan, Q. An overview of existing object-oriented database systems. In *Object-Oriented Databases with Applications to CASE, Networks, and VLSI CAD*, R. Gupta and E. Horowitz, (Eds.), Prentice-Hall, Englewood Cliffs, NJ, (1991), pp. 101-116.
58. Howard, J. H., Kazar, M. L., Menees, S. G., Nichols, D. A., Satyanarayanan, M., Sidebotham, R. N., and West, M. J. Scale and performance in a distributed file system. *ACM Trans. Computer Systems* 6, 1, (Feb. 1988), 51-81.
59. Hudson, S. E., and King, R. The Cactis project: Database support for software environments. *IEEE Trans. Softw. Eng.* 14, 6 (June 1988), 709-719.
60. Hudson, S. E., and King, R. Cactis: A self-adaptive, concurrent implementation of an object-oriented database management system. *ACM Trans. Database Syst.* 14, 3 (Sept. 1989), 291-321.
61. Hull, R., and King, R. Semantic database modeling: Survey, applications, and research issues. *ACM Computing Surveys* 19, 3 (Sept. 1987), 201-260.
62. IEEE. Special Issue on Multimedia Information Systems. *IEEE Computer* 24, 10, (Oct. 1991).
63. Jazayeri, M. Objects for distributed systems. *Proceedings of the ACM SIGPLAN Workshop on Object-Based Concurrent Programming* (San Diego, CA), G. Agha, P. Wegner, and A. Yonezawa, (Eds.), SIGPLAN Notices 24, 4 (April 1989), pp. 117-119.
64. Jeffay, K., Lin, J. K., Menges, J., Smith, F. D., and Smith, J. B. Architecture of the artifact-based collaboration system matrix. *Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW '92)*, CSCW Architectures, (1992), pp. 195-202.
65. Kacmar, C. J., Leggett, J., Schnase, J. L., and Boyle, C. Data management facilities of existing hypertext systems. Department of Computer Science Technical Report No. TAMU 88-018, Texas A&M University, College Station, TX, (1988).

66. Katz, R. H. Computer-aided design databases. In *New Directions for Database Systems*, G. Ariav and J. Clifford, (Eds.), Ablex Publishing Co., Norwood, NJ, (1986), 110-123.
67. Katz, R. H. Toward a unified framework for version modeling in engineering databases. *ACM Computing Surveys* 22, 4 (Dec. 1990), 375-408.
68. Kent, W. Limitations of record-oriented information models. *ACM Trans. Database Syst.* 4, 1 (March 1979), 107-131.
69. Kim, W., and Lochovsky, F. *Object-Oriented Concepts, Databases, and Applications*. ACM Press/Addison-Wesley, Inc., New York, NY, (1989).
70. Kistler, J. J., and Satyanarayanan, M. Disconnected operation in the Coda file system. *Operating Systems Review, Special Issue: Proceedings of the 13th ACM Symposium on Operating Systems Principles*, (Pacific Grove, CA), 25, 5, (1991), pp. 213-225.
71. Lange, D. A formal model for hypertext. *Proceedings of the NIST Hypertext Standardization Workshop*, National Institute of Standards and Technology, (Gaithersburg, MD), (Jan. 1990), pp. 145-166.
72. Lesk, M. What to do when there's too much information. *Proceedings of the Hypertext '89 Conference*, (Pittsburgh, PA), (Nov. 1989), pp. 305-318.
73. Lucarella, D. A model for hypertext-based information retrieval. *Hypertext: Concepts, Systems, and Applications. Proceedings of the European Conference on Hypertext*, (France), A. Rizk, N. Streitz, and J. Andre, (Eds.), Cambridge University Press, Cambridge, U.K., (Nov. 1990), pp. 81-94.
74. Mahler, A., and Lampen, A. Shape - A Software configuration management tool. *Proceedings of the International Workshop on Software Version and Configuration Control*, (Grassau, FRG), (Jan. 1988), pp. 228-243.
75. Maier, D., Stein, J., Otis, A., and Purdy, A. Development of an object-oriented DBMS. *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA)*, (Sept. 1986), pp. 472-482.
76. Maier, D. Making database systems fast enough for CAD applications. In *Object-Oriented Concepts, Databases, and Applications*, W. Kim and F. H. Lochovsky, (Eds.), Addison-Wesley Publishing Co., Reading, MA, (1989), pp. 573-582.
77. Malcom, K. C., Poltrock, S. E., and Schuler, D. Industrial strength hypermedia: Requirements for a large engineering enterprise. *Proceedings of the Third ACM Conference on Hypertext (Hypertext '91)*, (San Antonio, Texas), (December 1991), pp. 13-24.
78. Manola, F., and Dayal, U. PDM: An object-oriented data model. *Proceedings of the International Workshop on Object-Oriented Database Systems*, (Pacific Grove, CA), (Sept. 1986), pp. 18-25.
79. McLeod, D. A perspective on object-oriented and semantic database models and systems. In *Object-Oriented Databases with Applications to CASE, Networks, and VLSI CAD*, R. Gupta and E. Horowitz, (Eds.), Prentice-Hall, Englewood Cliffs, NJ, (1991), pp. 12-25.

80. Moline, J., Benigni, D., and Baronas, J., Eds. *Proceedings of the NIST Hypertext Standardization Workshop*, National Institute of Standards and Technology, (Gaithersburg, MD), (Jan. 1990).
81. Moss, J. E. B. Design of the Mneme persistent object store. *ACM Trans. Off. Inf. Syst.* 8, 2 (April 1990), 103-139.
82. Nelson, M. N., Welch, B. B., and Ousterhout, J. K. Caching in the Sprite network file system. *ACM Trans. Computer Systems* 6, 1, (Feb. 1988), 134-154.
83. Newcomb, S. R., Kipp, N. A., and Newcomb, V. T. The "HyTime" hypermedia/time-based document structuring language. *Comm. ACM* 34, 11, (Nov. 1991), 67-83.
84. Özsu, M.T., Dayal, U., and Valduriez, P., (Eds.). *Distributed object management*. Morgan Kaufman Publishing, San Mateo, CA, (1993).
85. Palaniappan, M., Yankelovich, N., and Sawtelle, M. Linking active anchors: A stage in the evolution of hypermedia. *Hypermedia* 2, 1 (Jan. 1990), 47-66.
86. Peckham, J., and Maryanski, F. Semantic data models. *ACM Computing Surveys* 20, 3 (Sept. 1988), 153-189.
87. Potter, W., and Trueblood, R. Traditional, semantic, and hyper-semantic approaches to data modeling. *IEEE Computer* 21, 6 (June 1988), 53-63.
88. Purdy, A., Schuchardt, B., and Maier, D. Integrating an object server with other worlds. In *Object-Oriented Databases with Applications to CASE, Networks, and VLSI CAD*, R. Gupta and E. Horowitz, (Eds.), Prentice-Hall, Englewood Cliffs, NJ, (1991), pp. 216-236.
89. Rizk, A., Streit, N., and Andre, J. (Eds.), *Hypertext: Concepts, Systems, and Applications. Proceedings of the European Conference on Hypertext*, (France), Cambridge University Press, Cambridge, U.K., (Nov. 1990).
90. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W. *Object-Oriented Modeling and System Design*. Prentice-Hall, Englewood Cliffs, NJ, (1990).
91. Schnase, J. L. HB2: A hyperbase management system for open, distributed hypermedia system architectures. Dissertation, Department of Computer Science, Texas A&M University, College Station, TX, (1992).
92. Schnase, J. L., Leggett, J. J., Hicks, D. L., and Szabo, R. L. Semantic data modeling of hypermedia associations. *ACM Trans. Inf. Syst.* 11, 1 (Jan. 1993), 27-50.
93. Schnase, J. L., Leggett, J. J., and Hicks, D. L. HB1: Design and implementation of a hyperbase management system. *Electronic Publishing: Origination, Dissemination and Design* 6, 2 (July 1993).
94. Schnase, J. L., Leggett, J. J., Hicks, D. L., Nürnberg, P. J., and Sánchez, J. A. SP2 - An open, extensible hypermedia system architecture. (Submitted to HICSS-27), (1993).

95. Schütt, H. A., and Streitz, N. A. Hyperbase: A hypermedia engine based on a relational database management system. In *Hypertext: Concepts, Systems, and Applications. Proceedings of the European Conference on Hypertext*, (France), A. Rizk, N. Streitz, and J. Andre, (Eds.), Cambridge University Press, Cambridge, U.K., (Nov. 1990), pp. 95–108.
96. Schütt, H., and Haake, J. M. Server support for cooperative hypermedia systems. *Proceedings of Hypermedia '93 Conference* (Zurich, Switzerland), (March 1993).
97. Segev, A., Ed. Special issue on directions for future database research and development. *SIGMOD Record* 19, 4, (Dec. 1990).
98. Senko, M. Information systems: Records, relations, sets, entities, and things. *Inf. Syst.* 1, (1975), 3-13.
99. Shackelford, D. E. Requirements document for the UNC Distributed Graph Service. Department of Computer Science Technical Report No. TR91-051, University of North Carolina at Chapel Hill, Chapel Hill, NC, (1991).
100. Shackelford, D. E. Implementation design document for the UNC Distributed Graph Storage System. Department of Computer Science Technical Report No. TR92-015, University of North Carolina at Chapel Hill, Chapel Hill, NC, (1992).
101. Shackelford, D. E., Smith, J. B., and Smith, F. D. The architecture and implementation of a distributed hypermedia storage system. *Proceedings of the Hypertext '93 Conference*, (Seattle, WA), (Nov. 1993).
102. Smith, J. B., and Smith, F. D. ABC: A hypermedia system for artifact-based collaboration. *Proceedings of the Third ACM Conference on Hypertext (Hypertext '91)* (San Antonio, TX), (Dec. 1991), pp. 179-192.
103. Smith, K., and Zdonik, S. Intermedia: A case study of the differences between relational and object-oriented database systems. *Proceedings of the ACM OOPSLA '87 Conference*, ACM, New York, (1987), pp. 452-465.
104. Stonebraker, M., and Kemnitz, G. The POSTGRES next-generation database management system. *Commun. ACM* 34, 2, (1991), 78-92.
105. Stotts, P. D., and Furuta R. Petri-net based hypertext: Document structure with browsing semantics. *ACM Trans. Off. Inf. Syst.*, 7, 1, (January 1989), 3-29.
106. Stotts, P. D., Furuta, R., and Ruiz, J. C. Hyperdocuments as automata: Trace-based browsing property verification. *Proceedings of the European Conference on Hypertext (ECHT '92)*, (Milan, Italy), (Nov. 1992), pp. 272-281.
107. Streitz, N. A., Haake, J., Hannemann, J., Lemke, A., Schuler, W., Schütt, H., and Thüring, M. SEPIA: A cooperative hypermedia authoring environment. *Proceedings of the European Conference on Hypertext (ECHT '92)*, (Milan, Italy), (Nov. 1992), pp. 11-22.
108. Sybase, Inc. *DB-Library Reference Manual, Release 4.2*. 6475 Christie Avenue, Emeryville, CA 94608, (May 1990).

109. Tompa, F. W. A data model for flexible hypertext database systems. *ACM Trans. Off. Inf. Syst.* 7, 1 (Jan. 1989), 85-100.
110. Tichy, W. F. RCS - A system for version control. *Software -- Practice and Experience* 15, 7, (July 1985), 637-654.
111. Watters, C., and Shepherd, M. A. A transient hypergraph-based model for data access. *ACM Trans. Inf. Syst.* 8, 2 (April 1990), 77-102.
112. Wiil, U. K. Design and implementation of a hyperbase. Department of Mathematics and Computer, Institute for Electronic Systems Technical Report No. IR 90-03, The University of Aalborg, Aalborg, Denmark, (1990).
113. Wiil, U. K., and Østerbye, K. Experiences with HyperBase – A multi-user back-end for hypertext applications with emphasis on collaboration support. Department of Mathematics and Computer, Institute for Electronic Systems Technical Report No. IR 90-38, The University of Aalborg, Aalborg, Denmark, (1990).
114. Wiil, U. K. Issues in the design of EHTS: A multiuser hypertext system for collaboration. *Proceedings of the Hawaii International Conference on System Sciences (HICSS-25)*, (Kauai, Hawaii), (Jan. 1992), pp. 629-639.
115. Wiil, U. K., and Leggett, J. J. Hyperform: Using extensibility to develop dynamic, open and distributed hypertext systems. *Proceedings of the European Conference on Hypertext (ECHT '92)*, (Milan, Italy), (Nov. 1992), pp. 251-261.
116. Wiil, U. K., and Leggett, J. J. Concurrency control in collaborative hypertext systems. *Proceedings of the Hypertext '93 Conference*, (Seattle, WA), (Nov. 1993).
117. Woelk, D., Kim, W., and Luther, W. An object-oriented approach to multimedia databases. *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, (Washington, DC), (May 1986), pp. 311-325.
118. Woelk, D., and Kim, W. Multimedia information management in an object-oriented database system. *Proceedings of the Thirteenth International Conference on Very Large Data Bases*, (Brighton, England), (1987), pp. 319-329.
119. Yankelovich, N., Haan, B., Meyrowitz, N., and Drucker, S. Intermedia: The concept and the construction of a seamless information environment. *Computer* 21, 1 (Jan. 1988), 81-96.
120. Zobel, J. Wilkinson, R., Thom, J., Mackie, E., Sacks-Davis, R., Kent, A., and Fuller, M. An architecture for hyperbase systems. *Proceedings of the First Australian Multi-Media Communications, Applications and Technology Workshop*, (1991), pp. 152-161.
121. Østerbye, K. Structural and cognitive problems in providing version control for hypertext. *Proceedings of the European Conference on Hypertext (ECHT '92)*, (Milan, Italy), (Nov. 1992), pp. 33-42.
122. Østerbye, K., Nørmark, K., and Jeppesen, H. M. Hyperstructure programming environments. Presented at the *Fifth Nordic Workshop on Programming Environment Research*. Obtainable through the author (1991).

## APPENDIX I - WORKSHOP PARTICIPANT ADDRESSES

Christopher Clifton  
Northwestern University  
Dept. of EECS  
2145 Sheridan Road  
Evanston, IL 60208-3118  
Phone: 708-491-7642  
Fax: 708-491-4455  
Email: clifton@eecs.nwu.edu

W. Bruce Croft  
Computer Science Department  
University of Massachusetts  
Amherst, MA 01003  
Phone: 413-545-0463  
Fax: 413-545-1249  
Email: croft@cs.umass.edu

Edward Fox  
Dept. of Computer Science  
VPI & SU  
562 McBryde Hall  
Blacksburg, VA 24061-0106  
Phone: 703-231-5113  
Fax: 703-231-6075  
Email: fox@fox.cs.vt.edu

Mark E. Frisse  
School of Medicine Library  
660 S. Euclid Avenue, Box 8132  
Washington University  
St. Louis, MO 63110  
Phone: 314-362-2773  
Fax: 314-362-0190  
Email: frisse@informatics.wustl.edu

Richard K. Furuta  
Department of Computer Science  
Texas A&M University  
College Station, TX 77843-3112  
Phone: 409-845-3839  
Fax: 409-847-8578  
Email: furuta@cs.tamu.edu

Anja Haake  
GMD-IPSI  
P. O. Box 104326, DolivoStr. 15  
D-6100 Darmstadt  
F. R. Germany  
Phone: 49 6151 869-929  
Fax: 49 6151 869-818  
Email: ahaake@ darmstadt.gmd.de

Hector J. Hernández  
Department of Computer Science  
New Mexico State University  
Las Cruces, NM 88003-0001  
Phone: 505-646-1206  
Fax: 505-646-6218  
Email: hector@nmsu.edu

David Hicks  
Department of Computer Science  
Texas A&M University  
College Station, TX 77843-3112  
Phone: 409-845-9980  
Fax: 409-847-8578  
Email: hicks@bush.cs.tamu.edu

Charles J. Kacmar  
Dept. of Computer Science, B-173  
203 Love Building  
Florida State University  
Tallahassee, Florida 32306-4019  
Phone: 904-644-9661  
Fax: 904-644-0058  
Email: kacmar@psi.cs.fsu.edu

Danny B. Lange  
IBM Tokyo Research Laboratory  
Computer Science Institute  
1623-14, Shimotsuruma, Yamato-shi  
Kanagawa-ken 242  
JAPAN  
Phone: +81-462-73-4656  
Fax: +81-462-74-4282  
Email: danny@acm.org

John J. Leggett  
Department of Computer Science  
Texas A&M University  
College Station, TX 77843-3112  
Phone: 409-845-0298  
Fax: 409-847-8578  
Email: leggett@bush.cs.tamu.edu

Raymond McCall  
University of Colorado  
Campus Box 314  
Boulder, CO 80309  
Phone: 303-492-2800  
Fax: 303-492-6163  
Email: mccall\_r@gold.colorado.edu

Don McCracken  
Knowledge Systems  
4488 Sardis Road  
Murrysville, PA 15668  
Phone: 412-733-8233  
Fax: 412-325-2656  
Email: dlm@centro.soar.cs.cmu.edu

Steven E. Poltrock  
Boeing Computer Services  
P. O. Box 24346  
MS 7L-64  
Seattle, WA 98124-0346  
Phone: 206-865-3270  
Fax: 206-865-2964  
Email: poltrock@atc.boeing.com

Laurence C. Rosenberg  
Director, Information Technology &  
Organizations Program  
IRIS -- Room 310  
National Science Foundation  
1800 G Street, N.W.  
Washington, DC 20550  
Phone: 202-357-9570  
Fax: 202-357-0320  
Email: lrosenbe@note.nsf.gov

## APPENDIX I - WORKSHOP PARTICIPANT ADDRESSES (continued)

John L. Schnase  
School of Medicine Library  
660 S. Euclid Avenue, Box 8132  
Washington University  
St. Louis, MO 63110  
Phone: 314-362-3117  
Fax: 314-362-0190  
Email: schnase@medicine.wustl.edu

Helge Schütt  
STEP  
Stürtz Electronic Publishing GmbH  
Technologiepark Würzburg-Rimpar  
Kettelerstrasse, D-97222 Rimpar  
Germany  
Phone: +49-9365-8062-0  
Fax: +49-9365-8062-66  
Email: hes@step.de

Douglas E. Shackelford  
CB #3175, Sitterson Hall  
Department of Computer Science  
University of North Carolina  
at Chapel Hill  
Chapel Hill, NC 27599-3175  
Phone: 919-962-1898  
Fax: 919-962-1799  
Email: shackelf@cs.unc.edu

John B. Smith  
CB #3175, Sitterson Hall  
Department of Computer Science  
University of North Carolina  
at Chapel Hill  
Chapel Hill, NC 27599-3175  
Phone: 919-962-1792  
Fax: 919-962-1799  
Email: jbs@cs.unc.edu

P. David Stotts  
CB #3175, Sitterson Hall  
Department of Computer Science  
University of North Carolina  
at Chapel Hill  
Chapel Hill, NC 27599-3175  
Phone: 919-962-1833  
Fax: 919-962-1799  
Email: stotts@cs.unc.edu

Uffe Kock Wiil  
Afd. for Mat og Dat  
Aalborg Universitetscenter  
Fredrik Bajers Vej 7E  
DK-9220 Aalborg  
Denmark  
Phone: 45 98 15 85 22  
Fax: 45 98 15 81 29  
Email: kock@iesd.auc.dk

Maria Zemankova  
Director, Database and  
Expert Systems Program  
IRIS -- Room 310  
National Science Foundation  
1800 G Street, N.W.  
Washington, DC 20550  
Phone: 202-357-9570  
Fax: 202-357-0320  
Email: mzemanko@note.nsf.gov

Kasper Østerbye  
Afd. for Mat og Dat  
Aalborg Universitetscenter  
Fredrik Bajers Vej 7E  
DK-9220 Aalborg  
Denmark  
Phone: +45 98 15 85 22  
Fax: +45 98 15 81 29  
Email: kasper@iesd.auc.dk

## **APPENDIX II - LIST OF INVITED PRESENTATIONS**

### **Models and Architectures**

Helge Schütt - The Cooperative Hypermedia Server  
John Leggett, John Schnase - The HB2 Hyperbase System  
John Smith, Doug Shackelford - The Distributed Graph Server  
Uffe Wiil - Hyperform: Data Model and Architecture

### **Node, Link and Structure Management**

P. David Stotts - Constraints for Object and Structure Integrity

### **Integration of Information Retrieval**

Mark Frisse - Data and People Issues  
Bruce Croft - Integration Issues

### **Version Control**

David Hicks - Version Control in Open Hypermedia Systems  
Anja Haake - CoVer: A Contextual Version Server for Hypertext Applications  
Kasper Østerbye - Versioning Issues in Hypertext and the Hyperpro Versioning Model

### **Concurrency Control, Transaction Management and Notification Control**

Uffe Wiil - Hyperform: Concurrency Control, Transaction Management and Notification Control

### **Impacts of Application Environments on Hyperbase Systems**

Ed Fox - Electronic Libraries  
Steve Poltrock - Large Engineering Enterprises