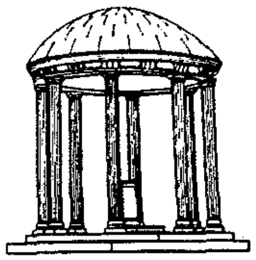
A Survey of Data Models for Hypermedia

TR91-022

April, 1991

Joan Boone

The University of North Carolina at Chapel Hill
Department of Computer Science
CB#3175, Sitterson Hall
Chapel Hill, NC 27599-3175
919-962-1792
jbs@cs.unc.edu



A TextLab Report
This work was supported in part by the IBM Corporation.
UNC is an Equal Opportunity/Affirmative Action Institution.

Abstract: This article surveys data models developed for hypermedia systems over the last several years. Each model is briefly summarized and identified according to the primary motivating factor: application requirements, refinement and formalization, and most recently, standardization. A more detailed description of the following data model characteristics is provided: low-level abstractions, high-level abstractions, operations, and relationship to system architecture. Lastly, the models are reviewed with respect to how they address some hypermedia issues such as versioning, collaboration, interoperability and interchange, and security.

Introduction

Hypermedia is an approach to the representation and management of information characterized by a set of nodes interconnected by links. Each node corresponds to a media object which may be text, graphics, audio, or video recordings; links represent relationships among media objects. Hypermedia research efforts and system implementations focus on what Conklin [Conk87] describes as either the front end (the user interface) or the back end (the database). The purpose of this article is to survey work related to an important component of the hypermedia back end: the data model. Data models are reviewed from several perspectives:

- *Motivating factors*

  Some of the earlier models were developed to support a specific application domain, e.g., software development or authoring environments. Others evolved as research projects, are more general purpose in nature, and serve to refine and formalize the hypermedia data model. Recent efforts propose reference models to promote standardization - an indication of the growing diversity of models appearing, and the need to support interchange among emerging systems.

- *Data model characteristics*

  Most models employ the basic node-and-link paradigm. This section focuses on the different refinements made to the basic model. Some of the properties that distinguish them are

  - Low-level representation primitives
  - Higher-level abstractions for aggregation and generalization
  - Operations to support model definition and manipulation
  - Relationship of the data model to system architecture, i.e., how does the data model interface with the underlying storage system and the user interface

- *Issues for hypermedia data models*

  Many of the identified data modeling issues are not unique to hypermedia; they are the same issues confronting modelers of conventional databases. However, given the collaborative and distributed nature of most hypermedia environments, these issues are especially important. This section reviews the issues and how some models propose to address them.

SOME PRELIMINARIES

What is a Data Model ?

Ullman [Ullm88] defines a data model as a mathematical formalism with two parts:

- o A notation for describing data, and
- o A set of operations used to manipulate that data

In the hypermedia domain the data model is typically some variation of the basic node-and-link paradigm. This model reflects the essence of hypermedia environments where information has a non-linear organization; it is stored in "chunks" with machine-supported links within and between the "chunks" [Conk87]. The notation used to describe the models varies from English language descriptions to more formal methods such as set notation and graph grammars. The set of operations provided are as diverse as the models themselves, but usually always include basic manipulation primitives for viewing and editing data objects. A summary of model abstractions and operations is provided in the "Data Model Characteristics" section.

Why do Hypermedia Systems Need a Data Model ?

There are two ways to answer this question:

- o From an *architecture* perspective it is important to isolate the data model as a separate layer in a hypermedia system for several reasons:
  - o So that diverse applications and presentation services can be built on a single hypermedia database.
  - o To maintain independence from the physical implementation; the data model should be independent of the underlying file management system to support system portability.
  - o Data models provide the definition mechanism to partition information in a systematic way for distributed data environments.

- o From a *database* perspective, the traditional models (hierarchical, network, relational) do not map well to the hypermedia environment for at least two reasons:
  - o Restrictive semantics of the entities and relationships in these models
  - o Lack of support for structural abstraction. The class of semantic models probably provides the best fit since it supports constructors for building complex data types [Kim89].

Most of the models in this review emphasize the structural abstractions not found in traditional models. The importance of positioning the data model within a system architecture has only recently been addressed by reference models developed by standardization work groups.

HISTORICAL BACKGROUND

The first description of hypertext is attributed to Vannevar Bush in his 1945 article "As We May Think" [Bush45]. Even the extended concept of hypermedia and the use of a persistent information store can be found in Bush's comments on the recording of research results:

> "A record, if it is to be useful to science, must be continuously extended, it must be stored, and above all it must be consulted. Today we make the record conventionally by writing and photography, followed by printing; but we also record on film, on waxdisks, and on magnetic wires."

Bush proposed the "memex," a device for the storage and retrieval of such records. Two essential capabilities of the "memex" continue to be fundamental properties of hypermedia data models:

- o "The process of tying two items together is the important thing."
  The basic node-and-link paradigm is common to most hypermedia models.
- o "Selection by association"
  As noted by Conklin, machine-supported links are the essence of hypertext that permit the traversal of a network of nodes in a user-directed manner.

In 1958 Good [Good58] proposed a network model of knowledge that he described as resembling Bush's "memex." This network contained "nodes and roads" - nodes which represent propositions, and roads which represent associations of varying strength. He compared the network to the nervous system where the network is highly connected, perhaps with some tree structures, but with "innumerable cross-connections." His model also introduced two abstractions:

- o The notion of changing views as one moves further from the network. For example, at close range it is a network of propositions, then a network of documents, then a university diagram.
- o The notion of aggregation where a subset of nodes is a "clump or ganglion" with a computable attribute known as "clumpiness."

Influenced by both Bush and Good, Engelbart wrote "A Conceptual Framework for the Augmentation of Man's Intellect" in 1963 [Enge63]. He proposed the H-LAM/T (Human using Language, Artifacts, and Methodology, in which he is Trained) concept where a hierarchically structured repertory of capabilities provides the foundation for augmenting intellect. Users work "within a symbol structure of some sort, shifting their attention from one structure to another as they guide and execute the processes that ultimately provide them with the comprehension and the problem solutions they seek." Several years later these concepts were implemented as NLS (oN Line System), a text processing system for a multi-user environment. The data model is essentially hierarchical where each node represents a "statement." A statement is the basic work unit which may represent paragraphs or sections. Links are supported for text citation and can be specified between and within nodes in the file hierarchy [Enge73]. NLS is now known as Augment and has been made available commercially.

Work Related to Hypermedia Data Models

OVERVIEW

This section provides an overview of data models that have been developed during the last several years. Earlier models were developed to support specific applications and were implemented as part of a larger system. Others have evolved as a means to refine and formalize the data model. Several of the most recent efforts address the need for model standardization in future hypermedia systems.

GRAS

GRAS (GRAph Storage) is the database component of a system developed at Osnabruck University to support software development environments [Bran85]. Software documents are represented as attributed, labelled *graphs* which are organized into *database* collections. Separation of structural and non-structural information is a central design principle which is realized in both the graph model and the storage representation to achieve an efficient implementation for associative queries.

KMS

KMS (Knowledge Management System) is a distributed hypermedia system for collaboration based on the ZOG system developed at Carnegie-Mellon University [Aksc88, Aksc84]. Unlike most hypermedia data models, the basic paradigm is tightly coupled with the user interface. The screen-sized *frame* is the primary object in KMS; users navigate among interconnected frames to view and edit their contents. Although frames are typically organized hierarchically, KMS permits the user to augment the structure with *links* for cross-referencing and commentary.

HAM

The HAM (Hypertext Abstract Machine) is a general purpose, transaction-based server for hypertext storage that is based on the storage system of Neptune, a prototype hypertext system developed at Tektronix Computer Research Laboratory [Camp88, Deli86]. The graph-based, object-oriented storage model employs hierarchically-organized *context* objects to partition *graph* contents. Contexts may contain *node* objects of arbitrary data related by *link* objects; relationships between nodes in different contexts are supported by cross-context links. In addition to model and storage management, the HAM provides versioning, filtering, and access control mechanisms.

HyperBase

HyperBase is a general purpose hypermedia engine that was motivated by the development of an authoring environment at the Integrated Publication and Information Systems Institute, West Germany [Schu90]. HyperBase uses a commercial relational database system that supplies transaction management and multi-user access services. Influenced by the HAM model, the HyperBase data model is object-oriented, and application and storage-system independent. *Complex objects* provide an abstraction capability to represent collections of references to *nodes* and *links*. This abstraction, as well as nodes and attributes, maintain history information about object modification. Complex objects additionally maintain versions which record the state of invalidated objects.

Graph Server

The Graph Server is a transaction-based, multi-user server for object-oriented hypertext applications under development at the University of North Carolina [Ande90]. The underlying data model is general purpose, graph-based, and object-oriented. Like the HAM's contexts, the *subgraph* object provides a useful abstraction for refining the graph database. The subgraph has extended semantics that include typing (e.g. directed graphs, connected graphs, lists, trees, acyclic graphs) and aggregation into composite subgraphs by bridging and embedding individual subgraphs.

MINOS

MINOS is a prototype multimedia information system developed at the University of Waterloo that supports presentation, browsing, extraction, sharing, editing, and formatting of multimedia documents [Chri86]. The document model is object-oriented with support for higher-level abstractions such as aggregations and generalizations. Links between objects are known as *annotations* that relate logical components. The document model actually consists of two submodels: the *logical model* describes the logical components of documents such as paragraphs or audio segments; the *physical model* defines the presentation specifications for the logical component. The submodels are united by mapping objects that relate the logical and physical components for information display.

Trellis

The Trellis hypertext model is the basis for a prototype hypertext browsing and authoring environment developed at the University of Maryland [Stot89]. The system is based on a Petri net model which clearly separates document structure, content, and presentation via mapping functions in the Petri net definition for a given hypertext. For example, content *elements* map to Petri net places, places map to *logical windows*, and transitions map to *logical buttons* that represent browsing actions. Unlike other graph-based models, the Petri net model permits not only the specification of information structure and content, but also the specification of browsing and concurrent execution semantics.

Intermedia

Intermedia is an object-oriented hypermedia system that supports applications development [Meyr86]. The system, developed at Brown University, has a single database built on the Ingres database management system for network-wide access and concurrency control. *Block*, *link*, and *web* objects are the foundation of the document-oriented data model and are the user paradigm for interaction with the database. Blocks are document segments, or anchors, which can be related through navigational links. Their definition is very open-ended, permitting the user to equate any valid selection from the user interface to a block. The abstraction for a collection of blocks and links is the web. By defining webs, users can access and share groups of documents based on various sets of linkages, thus providing different views of related documents.

The following models were primarily motivated by the need to refine and formalize the hypermedia data model versus supporting a specific implementation. As prototypes and systems have developed and been exercised, several design issues have been identified and addressed by these efforts.

Contexts

User collaboration is a primary objective of hypermedia systems. To be effective, Delisle and Schwartz [Deli87] identify several characteristics which must exist in a collaborative environment:

- o A means to organize related hypermedia information
- o Independent partitions to minimize interference among users
- o Versioning mechanism and a configuration facility to relate versions of nodes and links
- o Distributed database support

The authors propose *contexts* to partition hypermedia information into disjoint collections of nodes and links which may be related by cross-context links. An example of usage is document revision by multiple users. Each user defines a context for their work unit which may reside on different machines. The revised contexts are eventually merged back into the master context for the document, with appropriate updates made to the version histories of node and links.

Garg's Abstraction Mechanisms

Garg's work focuses on the importance of abstractions in the hypermedia model [Garg88]. In addition to aggregation and collaborative partitioning, Garg identifies several other ways that abstractions can be of value:

- o Filtering information based on its relevancy to the user
- o Definition of information structure versus content
- o Definition of domain knowledge rather than information instances
- o Maintenance of revisions

A set theoretical model is used to formally define three useful abstractions that permit the manipulation of information with different views and granularity.

- o *Aggregation* is a collection of objects that can be referenced by an identifier. Since an aggregate object is similar to a relation, operations such as projection, join, and selection can be defined.

- o *Generalization* is a collection of objects which share a similar characteristic. In contrast with an aggregation, individual properties of constituent objects are hidden; only shared characteristics are visible. Advantages of this view are the specification of generalized queries, attributes, and relationships, as well as default properties for the collection.
- o *Revision* is an information object that contains some changes that distinguish it from a source object. The concept of a revision tree is employed to define "delta sequences" of changes to an object. This approach permits the recreation of a previous version by applying the appropriate backward deltas to the current object.

A simple filtering mechanism can be applied to each of these abstractions through the specification of keyword attributes associated with objects and links. An interesting extension to this mechanism is suggested whereby the predicate meaning of a link can be interpreted by a PROLOG-like language for filtering information.

Hypergraph Data Model

The data model adopted by many hypermedia systems is based on a directed graph. Tompa [Tomp89] proposes a model based on directed, labelled hypergraphs to address several shortcomings of the simpler graph model:

- o Inadequate separation of nodes and content
- o Inadequate support for shared structures of components
- o No support for sets of pages, where pages represent node contents

A formal definition of the hypergraph data model includes a set of *nodes*, a set of *pages*, and a value function which maps nodes to pages, to achieve the desired separation. Sharing of structures among users is accomplished via *user views*. Unlike traditional database views which often are a restriction on database contents, Tompa's user views can augment or override the underlying database, thus also providing a customization capability. Labelled hyperedges provide the means for specifying sets of nodes and their associated pages.

The following models are not hypermedia-motivated but are of interest in that they provide additional formalisms for a graph-based representation of data. Both are oriented toward the development of database user interfaces.

Conceptual Graphs

Sowa proposes a formalism, *conceptual graphs*, to describe data in a way which is similar to a user's mental picture of that information [Sowa76]. The familiarity and naturalness of the paradigm are expected to facilitate the user's interaction with a database system without having to be knowledgeable about the underlying representation.

A conceptual graph is an undirected graph with two node types:

- o *Concepts* are the basic unit; they are essentially labelled symbols which can represent anything the user chooses
- o *Conceptual relations* are the connections between concepts and can contain one or more links

A powerful enhancement to the basic graph model is achieved by integrating artificial intelligence concepts. Formation rules and inference rules are used to maintain model integrity and to infer relations that are not explicitly defined. Relationship semantics are also enhanced via *function links* between concepts that permit the specification of quantifiers and functional dependencies. The resulting graph is called a *conceptual schema* which can map directly to a relational database.

GOOD

The GOOD (Graph-Oriented Object Database) model was developed to demonstrate the value of graphs to both describe and manage databases [Gyss90]. The basic model is a directed, labelled graph containing nodes that represent database objects, and interconnecting edges. The GOOD data manipulation language supports five operations: the addition and deletion of nodes and edges, and an abstraction operator that specifies a collection of nodes based on shared properties. An interesting technique employed by the operations is the use of a metamodel construct, the *pattern,* to describe subgraphs in a database instance. The pattern is the mechanism for selecting a subgraph for addition, deletion, or abstraction.

Many hypermedia systems have been motivated by front end objectives and implementation-specific requirements. These efforts have significantly advanced the technology but have at the same time resulted in a variety of highly disparate systems which cannot easily communicate with each other. Given the importance of collaboration and the inherent distributed nature of hypermedia information, this situation inhibits the effectiveness of existing and future systems. As a result, various proposals for hypermedia reference models have been made. These models serve several purposes:

- o Through formal descriptions they provide a basis for comparison and analysis
- o By developing standards for systems and terminology, standard interfaces for interchange and communication can be developed
- o They provide a framework within which design issues, problems, and solutions can be identified

The following models were presented at the Hypertext Standardization Workshop, sponsored by the National Institute of Standards and Technology in January, 1990.

Trellis Hypertext Reference Model

This reference model describes a three-level architectural framework that is primarily concerned with the presentation of hypertext information [Furu90]. The *concrete level* addresses the physical organization of the hypertext, i.e., how to format and display characteristics of abstract components. The *visible level* maps the concrete representation to the external user view. Below the concrete level is the *abstract level* of hypertext which defines its components and associations. Components include:

- o *Structure* - the "placeholders" and relationships which define the hypertext organization but are separate from the contents. This is the component which would include the data model; however,this reference model makes no assumptions about the organization of the underlying model. In their work the authors used the Petri Net model as described earlier.
- o *Contents* - text, graphics, audio or video segments
- o *Buttons* - definitions of how relationships are displayed
- o *Containers* - how to aggregate information for display purposes

At the abstract hypertext level associations between components are defined, i.e., the associations between structures and contents, buttons, and containers. Although this proposal provides minimal detail about the data model, it is included here because it provides a useful framework for the mapping of back end abstractions to front end abstractions.

Strawman Reference Model

This reference model describes hypermedia systems in terms of basic, advanced, and open features [Thom90]. *Basic* features are those which all systems have and can provide a foundation for comparison:

- o Media types - the content part of the representation of hypermedia information
- o Data model - the structure part of the representation. Like the Trellis model, this model does not define a specific paradigm, but does note that in all hypermedia systems the common primitive is the link object.
- o User interface
- o Persistent store

Advanced features are not found in all hypermedia systems but are generally required for more complex applications. These features include:

- o Multi-user support
- o Distributed data
- o Uniform representation - unlike existing models which emphasize separation of node contents and structure, this proposal suggests a recursive data model where contents can contain nodes, thus permitting structure within media objects.
- o Computational completeness - inclusion of procedural information in the data model to describe the behavioral characteristics of information
- o Specialized facilities for navigation, search, and complex queries
- o Versioning, configurations, and change management support

Open features are generic characteristics found in many computer systems and include human factors provisions, an open and modular architecture, portability and availability.

An ideal architecture for hypermedia systems that is characterized by the basic and advanced features is described. An important aspect of the architecture is its modularity which the authors argue will accelerate the development of system standards.

Van Dyke Parunak's Reference Model

The model outlined in this paper focuses on the functional elements, implementation, and interface issues of hypermedia systems [VanD89]. The functional elements address the basic components of the hypermedia model: nodes and links, and the composite structures derived from them. The salient characteristics of the elements are described:

nodes can be characterized by their contents, type, and structure; links can be characterized by their directionality, topology, type, anchors, and modes. Composites can be defined either topologically or rhetorically. An example of a topological composite is a predefined route for navigation through an application. Rhetorical composites are logical groupings of nodes and links which, for example, might represent an argumentation schema.

The functional elements are the foundation of a four layer architecture for hypermedia implementation. At the lowest level is the data layer which is responsible for data and transaction management in a multiuser environment, distributed data access, and possibly versioning. The next higher level is the element layer that supports node and link management services. Generalized link traversal is provided by the inference layer, and the user's view is managed by the interface layer. Several user interface issues are identified which deal primarily with techniques for constructing links, screen layout and manipulation, and navigational mechanisms.

Dexter Reference Model

Unlike the previous models described here, the Dexter model provides a more extensive and formal definition of meaningful hypermedia abstractions [Hala89]. These are part of the *storage layer*, the focus of the three-layer reference model. The storage layer is the middle "database" layer which describes the network of components and interconnecting links. Components are the basic unit of the model and represent generic placeholders for hypermedia contents. A component may represent an atom (similar to a node), a link, or a composite which consists of other components.

As in other models, separation of structure and content is an important objective and is maintained by associating contents with the *within-component layer*. Component contents (text, graphics, audio, etc.) and internal structure are not part of this model; however, a generic interface between the storage and within-component layers is provided by an anchoring mechanism. Anchors define link end points and may be associated with a segment of a component's contents, e.g., a document paragraph or citation.

The third layer of the model is the *runtime layer* which supports the presentation of hypermedia information. Like the within-component layer, the runtime layer is not addressed by this model in detail; instead, the emphasis is on the storage layer interface to the presentation services. A generic interface is supplied via presentation specifications that describe how a component is to be displayed to the user. The specifications are included in the storage layer definition of a component and are made accessible to the presentation services.

In addition to the structural definition provided by the storage layer, the Dexter model defines a set of operations to access and manage hypermedia components. These include the basic management primitives for addition, deletion, and modification, and a pair of resolver/accessor functions for retrieval. A formal specification of the operations and abstractions is included in the model.

Lange's Formal Model

Lange's hypertext model concentrates primarily on the data model; it is not an architectural framework for hypertext systems [Lang90]. *Nodes* and *links* are the basic units of the model which defines two composite abstractions, *networks* and *structures*. Unlike the previous models, Lange's does not strongly separate structure and content, but rather provides explicit support for contents. His model defines the *slot* abstraction as a node substructure, or template, for contents. A node may contain multiple slots that are connection points for links. Slots are further refined through the notion of *buttons* and *fields* which are anchoring points for links and represent text segments in a document.

Networks are an abstraction for a set of hypertext links. *Structures* define an organization for nodes and networks; for example, a structure might be a sequence, a set, or a tree, depending on the application.

DATA MODEL CHARACTERISTICS

Despite the diversity of purpose, refinement, and implementation, the data models described here have several common characteristics: each defines primitive objects which are interrelated in some way and each has a basic set of operations for viewing and editing objects. Each model also defines at least one higher level abstraction which serves as a collector for primitive objects. The following table summarizes the structural abstractions and operations that characterize each model.

Table 1. Summary of Data Model Characteristics

DATA MODEL	LOW LEVEL ABSTRACTIONS	HIGH LEVEL ABSTRACTIONS	OPERATIONS
GRAS	attributed nodes, labelled edges	attributed and directed graphs, database, multi-database	insertion, deletion, assignment, query
KMS	frames, links	hierarchy of frames	navigation, frame editing
HAM	nodes, links,cross-context links, attributes	graphs, contexts	editing, filtering, string search, merge
HyperBase	node and link objects with attributes	complex objects	editing, copy
Graph Server	nodes, links, attributes	database, subgraph, composite subgraph (bridged and embedded subgraphs)	editing, copy, traversal, set operations on subgraphs
MINOS	attributed objects, annotations	aggregation hierarchies, generalization hierarchies	viewing, browsing; information extraction, sharing, correlation, generation; document formatting
Trellis	places, transitions, flow relations	Petri net	browsing, editing
Intermedia	blocks and links with keywords	documents, webs	generic management operations for blocks, links, webs and their attributes
Contexts	nodes, links, cross-context links	context, supercontext	editing, navigation, pruning, destroy, merge, difference detection
Garg's Abstraction Mechanisms	objects, attributes, predicates	aggregation, generalization, revision	filtering
Hypergraph	nodes, hyperedges, labelled edges	user views	query, view creation, update

Conceptual Graphs	concepts, concept relations	conceptual graphs, conceptual schema	copy, detach,restrict, join, conjunction,negation, disjunction,implication
GOOD	nodes, edges, labels	directed and labelled graphs, subgraphs	add, delete, abstract, query, browse, restructure, update
Trellis Reference Model	placeholders, relationships	abstract structure	
Strawman Reference Model	nodes, links		
Van Dyke Parunak's Reference Model	nodes, links	composites	
Dexter Reference Model	components (atoms, links), anchors, attributes	composite components	addition, deletion, modification, and retrieval of components
Lange's Formal Model	nodes, links, slots, buttons, fields	structures, networks	generic management operations for node, networks, and structures; version management, access control

Low Level Abstractions The low level abstractions in most models are structural entities that are independent of their content, applications, and the user environment. Two exceptions are KMS where nodes are represented by screen-sized workspaces called frames and the structured definition of node contents in Lange's model. His node interior definition includes a collection of uniquely-identified slots which in turn may contain buttons and fields that are anchorable locations for links.

The most common refinement for nodes and links is the specification of attribute-value pairs which may be system- or user-defined. The Petri net is an exception in that nodes are viewed as placeholders only, without any attributes.

Link definitions exhibit greater diversity across the models than node definitions. Several distinguishing characteristics are type, direction, arity, attributes, anchors, and relationship to nodes. Most models treat links as "first-class" objects, i.e., they are abstractions whose semantic importance is equivalent to nodes. In several models link-specific high level abstractions have been defined which provide a "collection of links" perspective: Hypergraph defines hyperedges, Intermedia has webs, and Lange defines networks. Garg, on the other hand, does not explicitly define links in his abstractions, but rather refers to "two-place predicates" on objects. The Trellis Petri net model views links as transitions to define browsing execution, as opposed to the more conventional view of links as structural connectors.

In most models links are binary and without type; however, the Dexter model permits links of arbitrary arity, and the KMS distinguishes between structural links (tree items) and referential links (annotation items). An important refinement of the link construct is the anchor, generally defined as the end point of a link. The anchor references some portion of a node's contents and is used, for example, to relate text segments within and between documents. The anchor is considered a structural entity whose value is managed by the application so as to maintain the separation between structure and content. Like other model constructs, the anchor has various refinements. The Dexter model provides the most formal definition which includes an anchor id and value that are part of the component specifications; the anchor value is arbitrary and application-defined. Lange distinguishes between the source and target end points of a link - these are known as anchors and destinations, respectively. A link is anchored to buttons and fields in a node and may have multiple endpoints associated with it. HyperBase describes "point-to-point links" which are functionally similar to anchored links but are not supported by an explicit anchor abstraction. It is suggested that object attributes can be used for this purpose. Van Dyke Parunak discusses link end points between structured and unstructured nodes and describes two types of end points: one which references a node substructure, the other references an arbitrary segment of node contents.

High Level Abstractions Another characteristic shared by the hypermedia data models is the provision for a higher level abstraction, usually of an aggregation type. For the graph-based data models this abstraction is a graph or subgraph. GRAS employs attributed, directed graphs which can be grouped together to form databases. In the HAM storage model the graph is the highest level object which is partitioned into contexts. The Petri net abstraction in the Trellis reference model is a generalization of a directed graph that also defines the browsing semantics for the user interface.

A more complete refinement of the graph abstraction can be found in the Graph Server. In this model subgraphs represent a set of nodes and links and are the basic unit of manipulation, i.e., all operations on nodes and links are performed in the subgraph context. Subgraphs are typed, and correctness is ensured through type-specific methods. Another level of abstraction is provided by composite subgraphs which are formed by bridging and embedding graphs. A *bridge graph* represents a set of links which spans multiple subgraph instances and the set of incident nodes. An *embedded graph* is a bridge graph where the bridge links are restricted to relating nodes within the same subgraph.

The MINOS and HyperBase systems adopt an object-oriented view of the data model. MINOS defines aggregation and generalization hierarchies of objects, whereas HyperBase defines the more general complex object as a collection of objects which may optionally be ordered. Garg's abstraction mechanisms include revision since a most current version "conceals" historical version information. This is similar in concept to Delisle and Schwartz's "contexts" which can be used to partition a document's contents for revision by multiple users.

Operations Operations for most of the models described here include specifications for generic management functions such as viewing and editing. To varying degrees the operations ensure model integrity. The Graph Server verifies type-correctness of subgraphs; HyperBase prohibits dangling references, ensures object uniqueness, and disallows recursive definition of complex objects. An important operation in the Contexts model is the merge. This model provides support for detecting conflicts and highlighting differences when a revision is merged with the original context.

Relationship to System Architecture A general architecture which characterizes most computer systems consists of three layers: end user interface, applications, and data management. The systems and models described here emphasize a refinement of the data management layer that distinguishes between the structure and content of information. This approach is especially important for hypermedia systems where the same structure may be used to represent a variety of media types. Conversely, the same media types may map to multiple structures to support different user views or display devices. Several methods have been used to maintain data model independence from contents and the end user interface.

Both the Trellis and Hypergraph models provide formal definitions of the relationship between structure and content. The Trellis Petri-net model defines a mapping function from content elements to "places" in the net. A similar function can be found in the hypergraph definition which relates nodes with pages (or contents). The MINOS implementation employs logical tables which define the aggregation hierarchies that map to the document files. The Dexter model defines a clear separation via its layered architecture. The storage layer contains the structural definition of hypermedia components which interfaces to the within-component layer through anchoring specifications.

Data model support for the end user interface is another discriminating characteristic. Although most models are concerned with data representation, the importance of coupling the hypermedia front end with the back end is reflected in some models. In the MINOS model the physical table defines the presentation characteristics of components in the logical table. A mapping mechanism between the tables permits multiple presentation schemes for a set of logical components. Presentation specifications in the Dexter model are included in component definitions, thus providing a generic mechanism for interpretation by the presentation services.

Perhaps the tightest coupling of model with user interface can be found in the Trellis reference model where the visible and concrete layers of the model define the format and display of hypertext information. The abstract layer, which encompasses the data model, emphasizes the association of structures with buttons (how relationships are displayed) and containers (how to display aggregates). This close coupling of user view with data view results in a consistent system architecture based on the node-and-link paradigm that is effective in a single-user, single-database environment. Whether this same approach is compatible with interoperability objectives is an issue to be evaluated.

Data Model Issues

As hypermedia data models mature, design issues are emerging which have been addressed to varying degrees by the models included here. The issues are not hypermedia-specific, as they apply to any database environment; however, they are particularly significant for the development of collaborative and distributed hypermedia systems.

Versioning

Versioning support provides the capability to manage changes to information over time; this contrasts with traditional "snapshot" databases where the state of an object has one representation which is subject to modification. An extension to versioning is the concept of configuration where a collection of versions of related information is maintained. For example, a software configuration management system maintains multiple versions of source code, which is related to multiple versions of documentation and object modules.

Some of the versioning issues which affect data modeling include:

- o Application versus data model support

  Responsibility for defining, maintaining, and validating version control mechanisms varies widely. Functionality that was once the domain of application programs is now being absorbed by semantic data models.
- o Version propagation

  When a component of a composite object is revised, is version information propagated to related components.?
- o Space efficiency

  Should new versions be represented as copies or deltas to the original object ?
- o Configuration integrity

  How can the consistency and completeness of a configuration be verified ?

The Strawman reference model assumes no versioning support is provided by the data model. The architecture includes a change management module that handles the recording and propagation of changes. The HyperBase engine provides some definition support for history information, but assumes the application interface supplies the management function. Model support includes definition of history attributes for individual objects and maintenance of versions for complex objects. The KMS supports versioning of frame hierarchies by maintaining linked lists of successive versions.

The Neptune model, on which the HAM is based, maintains complete version histories of graphs. Two link types are defined: one which refers to a specific version, the other to the most current version. This typing can be used to define a configuration by designating a "configuration node" with links to related versions. Concerned with space efficiency , the Neptune model records version changes as deltas to the original, not with multiple copies. This approach is very similar to the functionality outlined by Garg for the revision abstraction.

MINOS and Contexts both support the notion of version trees; i.e., the tree root represents the original document, and the leaf nodes correspond, for example, to annotated versions generated by reviewers. MINOS improves on space requirements by permitting the sharing of common objects. Contexts supports sophisticated merge operations to consolidate multiple revisions into a new document version. The multi-user and distributed characteristics of Contexts' versioning most nearly approaches the requirements of collaborative environments.

Collaboration

Support for collaborative environments by hypermedia systems exhibits the same features and problems as very large database environments: multiple users, concurrency control, and distributed function. The data model implications of this environment have not been investigated extensively; however, concerns on how to provide support have been expressed, and partial solutions proposed.

Perhaps the most expeditious approach was that taken by HyperBase which is "tightly coupled" with a commercial data base management system to provide concurrency control and transaction management. Two of the reference model proposals suggest organizations which emphasize the role of the data model for distributed support. Van Dyke Parunak's four-layer architecture isolates the data layer which would provide distributed data access. Thompson's Strawman model suggests that distributed support could be implemented with a relational database for structural information and a WORM device for contents.

Concurrency is improved in the KMS data model by defining a construct that represents a small unit of work. The frame is the basic unit and typically corresponds to a few paragraphs. The small unit, in conjunction with an "optimistic" concurrency control algorithm, reduces interference among users when manipulating large amounts of data. Tompa's hypergraph model also defines a unit of work, the "user view," which is the basis for update and managing concurrency.

The most extensive work thus far in developing a useful data model for distributed, collaborative environments is Delisle and Schwartz's Contexts. Their data partitioning concept, which has been implemented in the HAM, addresses several needs: a useful aggregation abstraction for nodes and links, a mechanism for defining independent partitions to reduce user interference, and a mechanism for defining distributed units of work.

Interoperability and Interchange

An issue related to distributed hypermedia environments is the need to communicate among heterogeneous hypermedia systems. The ideal solution would be one where a global schema maps to, and resolves, the differences among various systems, thus allowing applications to access any environment with the same interface.

An interim solution which provides a measure of interoperability is Sun's Link Service [Pear89]. The service defines a protocol which allows independent applications to define relationships in a hypertext system. The relationships are stored as pointer pairs between linked objects in a link database. By registering with the link database, an application becomes part of an extensible front end to the hypertext system.

In the absence of a global schema or interoperability, exchange of information between heterogeneous systems requires file format standardization. Riley's proposal for an interchange format standard supports "first-order" hypertext systems, i.e., those which define documents, links, anchors, and attributes [Rile90].

Security

Restricting access to information is a concern in any multi-user environment. Security mechanisms are found throughout a system, from the user interface to the underlying file system. The types of restrictions and the granularity of data to which restrictions apply can vary greatly.

For database security the unit of access is often derived from the data model representation. For example, the basic unit of the KMS model is the frame. Each frame has an owner and access control is obtained by owner-specification of permissions granted to other users. The Petri net model supports "subhypertexts" which are individual Petri net structures that have access control classes associated with them; the classes identify users with browsing capabilities. In the HAM, object types are optionally related to access control lists which define a user or group and associated permissions, e.g., access, annotate, update, destroy.

These methods assume a multi-user, single database environment. Additional work is needed to determine if they are adequate for the collaborative environment, and to what extent the data model provides support for security of hypermedia information.

Conclusion

In this paper I have reviewed the work related to hypermedia data models with the objective of consolidating and summarizing the developments in this important component of hypermedia systems. Where early models focused on satisfying requirements for a specific implementation, more recent efforts are concerned with refinement and standardization. These efforts have surfaced many issues, several of which are described here. Additionally, there are system-level issues identified by Halasz (Hala88] which have ramifications for data models of next-generation hypermedia systems. These include query-based access, composite and virtual structures to augment the basic node-and-link paradigm, extensibility and tailorability, and support for computational engines by integrating hypermedia with AI technology. Progress in these areas is important if hypermedia systems are to overcome a fundamental obstacle described by Bush in 1945: "we can enormously extend the record; yet even in its present bulk we can hardly consult it." Further refinements to existing data models will be essential for users to effectively use the abundance of information made available by future hypermedia systems.

REFERENCES

Aksc84	R.M. Akscyn and D.L. McCracken, The ZOG Approach to Database Management, *Proceedings of the 1984 Trends and Applications Conference: Making Database Work,* Gaithersburg, Maryland, May 1984.
Aksc88	R.M. Akscyn, D.L. McCracken, and E.A. Yoder, KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations, *Communications of the ACM,* Vol. 31, No. 7, 820-835, July 1988.
Ande90	M. Anderegg, B. Elledge, J. Harford, D. Shackelford, and O. Toki, User's Manual for Attributed Directed Graph Server, Department of Computer Science, University of North Carolina, May 1990.
Bran85	T. Brandes and C. Lewerentz, GRAS: A Non-Standard Data Base System within a Software Development Environment, *GTE Workshop on Software Engineering Environments for Programming-in-the-Large,* Harwichport, Massachusetts, 113-121, June 1985.
Bush45	V. Bush, As We May Think, *Atlantic Monthly,*Vol. 176, No. 1, 101-108, July 1945.
Camp88	B. Campbell and J.M. Goodman, HAM: A General Purpose Hypertext Abstract Machine, *Communications of the ACM,* Vol. 31, No. 7, 856-861, July 1988.
Chri86	S. Christodoulakis, M. Theodoridou, F. Ho, M. Papa, and A. Pathria, Multimedia Document Presentation, Information Extraction, and Document Formation in MINOS: A Model and a System, *ACM Transactions on Office Information Systems,* Vol. 4, No. 4, 345-383, October, 1986.
Conk87	J. Conklin, Hypertext: An Introduction and Survey, *IEEE Computer,* Vol. 20, No. 9, 17-41, September 1987.
Deli86	N. Delisle and M. Schwartz, Neptune: a Hypertext System for CAD Applications, CR-85-50, Tektronix Computer Research Laboratory, Beaverton, Oregon, January 1986.

Deli87 N. Delisle and M. Schwartz, Contexts - A Partitioning Concept for Hypertext, *ACM Transactions on Office Information Systems,* Vol. 5, No. 2, 168-186, April 1987.

Enge63 D.C. Engelbart, A Conceptual Framework for the Augmentation of Man's Intellect, *Vistas in Information Handling*, P.D. Howerton and D.C. Weeks (editors), Spartan Books, Washington, D.C., 1-29, 1963.

Enge73 D.C. Engelbart, R. W. Watson, and J. C. Norton, The Augmented Knowledge Workshop, *AFIPS Conference Proceedings, 1973 National Computer Conference and Exposition* (June 4-8, 1973, New York, New York), 9-21, 1973.

Furu90 R. Furuta and P.D. Stotts, The Trellis Hypertext Reference Model, *Proceedings of the Hypertext Standardization Workshop* (Gaithersburg, Maryland), 83-93, January 1990.

Garg88 P. K. Garg, Abstraction Mechanisms in Hypertext, *Communications of the ACM,* Vol. 31, No. 7, 862-870, July 1988.

Good58 I.J. Good, How Much Science Can You Have at Your Fingertips?, *IBM Journal,* 282-288, October 1958.

Gyss90 M. Gyssens, J. Paredaens, and D. Van Gucht, A Graph-Oriented Object Model for Database End-User Interfaces, *Proceedings of 1990 ACM SIGMOD International Conference on Management of Data* (May 23-25, Atlantic City,New Jersey), 24-33, 1990.

Hala88 F. Halasz, Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems, *Communications of the ACM*, Vol. 31, No. 7, 836-852, July 1988.

Hala89 F. Halasz and M. Schwartz, The Dexter Hypertext Reference Model, *Proceedings of the Hypertext Standardization Workshop* (Gaithersburg, Maryland), 1-39, January 1990.

Kim89 W. Kim and F.H. Lochovsky (editors), *Object-Oriented Concepts, Databases, and Applications,* ACM Press, New York, 1989.

Lang90	D. Lange, A Formal Model of Hypertext, *Proceedings of the Hypertext Standardization Workshop,* (Gaithersburg, Maryland), 145-166, January 1990.
Meyr86	N. Meyrowitz, Intermedia: The Architecture and Construction of an Object-Oriented Hypermedia System and Applications Framework, *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA '86)* (September 29-October 2, Portland, Oregon), 1986.
Pear89	A.Pearl, Sun's Link Service: A Protocol for Open Linking, *Hypertext '89 Proceedings* (November 5-7, Pittsburgh, Pennsylvania), 137-146, 1989.
Rile90	V. Riley, An Interchange Format for Hypertext Systems: the Intermedia Model, *Proceedings of the Hypertext Standardization Workshop* (Gaithersburg, Maryland), 213-222, January 1990.
Schu90	H. Schutt and N. Streitz, HyperBase: A Hypermedia Engine Based on a Relational Database Management System, Integrated Publication and Information Systems Institute, West Germany, submitted to ECHT '90.
Sowa76	J. Sowa, Conceptual Graphs for a Data Base Interface, *IBM Journal of Research and Development,* 336-357, July 1976.
Stot89	P. Stotts and R. Furuta, Petri-Net-Based Hypertext: Document Structure with Browsing Semantics, *ACM Transactions on Information Systems,* Vol. 7, No. 1, 3-29, January 1989.
Thom90	C. Thompson, Strawman Reference Model for Hypermedia Systems, *Proceedings of the Hypertext Standardization Workshop,* (Gaithersburg, Maryland), 223-246, January 1990.
Tomp89	F. Tompa, A Data Model for Flexible Hypertext Database Systems, *ACM Transactions on Information Systems,* Vol. 7, No. 1, 85-100, January 1989.
Ullm88	J. D. Ullman, *Principles of Database and Knowledge-Base Systems,* Vol. 1, Computer Science Press, Rockville, Maryland, 1988.

VanD89 H. Van Dyke Parunak, Toward a Reference Model for Hypermedia, *Proceedings of the Hypertext Standardization Workshop* (Gaithersburg, Maryland), 197-211, January 1990.