Uniform Comparison of Data Models Using Containment Modeling

E. James Whitehead, Jr.
University of California, Santa Cruz
Dept. of Computer Science
Santa Cruz, CA 95064
+1.831.459.1227

ejw@cs.ucsc.edu

ABSTRACT

Containment data models are a subset of entity relationship models in which the allowed relationships are either a type of containment, storage, or inheritance. This paper describes containment relationships, and containment data models, applying them to model a broad range of monolithic, link server, and hyperbase systems, as well as the Dexter reference model, and the WWW with WebDAV extensions. A key quality of containment data models is their ability to model systems uniformly, allowing a broad range of systems to be compared consistently.

Categories and Subject Descriptors

H.2.1 [**Database Management**]: Logical design – *data models*; H.2.3 [**Database Management**]: Languages – *data description languages;* H.3.4 [**Information Storage and Retrieval**]: Systems and Software

General Terms

Design, Theory

Keywords

Containment data modeling. Hypertext data models.

1. INTRODUCTION

In our daily lives, we use containers all the time. Students use backpacks or bags to carry their books, and travelers use suitcases to carry their clothes. When shopping, we place our purchases into a basket or cart, and then carry home the goods in a shopping bag. If we drove to the store, these bags are placed in the trunk of our car, making nested containers: goods in bag in trunk in car. In all of these examples, the item is physically contained within the container, and can only belong to one container at a time.

Unlike physical items, objects in a computer have the quality of easy duplication at low to trivial cost, and this means that computer containment is not zero-sum: the same object can belong to multiple containers. The ease of object duplication afforded by computers dramatically increases the utility of containing objects using references, and holding the same object in multiple containers.

Hypertext '02, June 11-15, 2002, College Park, MD.

Computer containers fill many roles, providing organization of large collections of objects into smaller units, a form of modularization [8], and information hiding via encapsulation [15,29]. Containers can also be used to model compound documents, for example, the combination of some text and image objects to model a document containing figures. Dexter composites exemplify this use [17]. Just as with physical containers, computer containers are used to transport items, examples including ZIP files, and the MIME multipart/related packaging of documents in electronic mail [21].

It is not surprising that containers frequently appear in computer information systems as a mechanism for grouping and organizing data items. What is surprising is the lack of emphasis on modeling these commonly occurring containment relationships. Ideally, we would like to model the containment properties of hypertext systems with a mechanism that has the following properties:

- *Uniformity:* model containment properties of systems using a minimal set of abstractions with constant meaning. Instead of providing a normative definition of hypertext concepts, and then mapping system abstractions into these concepts (a la Dexter [17]), ideally we want to use atomic entities to build up models of each system's hypertext concepts.
- *Utility:* easily answer basic hypermedia data model questions such as, "are links to whole works, or to a subregion in a work (i.e., is there a notion of anchoring?)", "are links separate from, or part of documents", and "is there some notion of composite?"
- *Support Analysis:* be able to examine systems to tease out different design spaces employed in each system's data model.
- *Graphic formalism:* Communicate to a wide range of parties, inside and outside of hypermedia community, with small time needed to learn graphic representation, and with most people having an intuitive understanding of the formalism. Allow commonality among systems to be visually evident.
- *Concise format:* Be able to fit the containment models of multiple systems onto a single page or screen. This allows more rapid comparison of system data models.
- *Cross-discipline:* Can be used to model the containment properties of other types of information systems, such as Software Configuration Management and Document

Management systems, and compare them to hypertext systems.

Previous work by the author introduced the concept of containment data modeling, using it to describe the design spaces for link and structure versioning that occur in hypertext versioning systems [34]. The present paper significantly extends this prior work by expanding the treatment of containment data modeling. It then proceeds to give containment data models of a wide range of hypertext systems, including examples of monolithic, link server, hyperbase, open hyperbase, and hypertext versioning systems, as well as the Dexter model [17], and WWW/WebDAV [35]. Common features visible in the data models of each class of systems, and across all systems, will be discussed.

2. CONTAINMENT DATA MODELING

Modeling systems in a uniform way requires that the modeling mechanism contain system-neutral basic modeling blocks. For example, the links in the Dexter hypertext reference model [17] could not be used as the basic modeling block for representing links in all hypertext systems, since Dexter links embody specific design choices (links contain two endpoints, where each endpoint holds a component identifier and an anchor identifier, in the context of a data model where the anchor is an attribute of the component) that are not present in other systems. Neither KMS [2] nor NoteCards [31] has a first-class anchor concept, and mapping their links onto Dexter links requires creating pseudo-anchors that are not present in the original system.

Ideally we want to create data models using non-hypertextual entities that can be mapped onto hypertext system abstractions in a direct and unforced way. To achieve this goal, extended entity-relationship models [7], an important member of the class of semantic data models [26,19], are the basic modeling method used for containment data modeling of hypertext systems. Since the notion of an entity carries with it relatively few assumptions, it carries relatively few biases into the model. This generality of the entity concept is what leads containment data modeling to have the property of uniformity when modeling hypertext systems.

A desire to avoid model bias motivates the choice of modeling entities using each system's specific terminology. It is possible to choose an abstract term such as "work" for the documents or media that are linked together, and then map onto it similar terms like document, node, component, and resource. This modeling move would privilege the chosen definition of work, and thus hide the differences in meaning inherent in the different terms. It would also make it more difficult to check the correctness of a model, since the correspondence between model entities and entities in the original source(s) is no longer one to one. However, using a single set of terms might improve modeling uniformity, and ease detection of patterns within the data models. For the present audience of hypertext researchers, we choose to model systems using their original terms; for future, more general audiences we will likely reverse the decision.

Entity-relationship modeling was chosen over alternatives such as object-oriented modeling [6] due to its emphasis on static relationships, and the fact that it does not involve modeling behavioral aspects of systems entities, such as methods and their parameters. Introducing methods and their parameters into the model acts, in this case, only to obscure the key containment and storage relationships. In the past, semantic data modeling using an enhanced entity-relationship model was successfully employed in developing the HB1/SP1 system [28].

The essential elements of entity-relationship data models are entities, and relationships [26,19]. Entities signify abstractions, such as works (documents), anchors, links, and container types. Typed relationships exist between the entities, and this type is either predefined, such as the "is-a" (inheritance) relationship, or is defined by a specific model. Containment relationships are an example of these. Graphical representations of entity-relationship data models can be made using the intuitive notion that entities correspond to nodes in a graph, while the relationships correspond to arcs. This is the same intuition that underlies viewing a hypertext as a graph, with works as nodes, and links as relationships.

With any modeling mechanism, some elements are emphasized, while others are abstracted away. Containment data modeling emphasizes containment and storage relationships among hypertext data model entities. Other important aspects of the internals hypertext systems are not visible in these models, including such items as scripting, search functionality, and message passing. These additional aspects can be depicted in other kinds of diagrams, requiring multiple diagrams to capture all facets of a particular system.

2.1 Modeling Primitives

2.1.1 Entities

A typical entity-relationship (E-R) model uses entities to model data items. In the traditional use of entity-relationship modeling for databases, entities contain a series of attributes, and these attributes are basic data types, such as integers, floating point numbers, and strings. For example, an address entity would be represented as containing street, city, and zip code string attributes. When modeling hypertext systems, entities represent abstractions such as works, anchors, and links. While the concrete representation of anchors and links is similar in granularity to typical entities used in database modeling, the concrete representation of works is much larger, and can be organized according to one of many different internal formats, such as a word processing, spreadsheet, or bitmap image organization. Objects, which represent works, anchors, and links, typically take one of three organizations: all data, data plus properties, and all properties. The data plus properties and all properties organizations are examples of data aggregation, where the object is composed of one or more properties, and, in the case of the data plus properties organization, a data item representing the contents. Departing from typical E-R diagramming convention, this aggregation of data items is not modeled by having the properties and contents be modeled as attributes. Instead, properties and contents are modeled as entities, and an inclusion containment relationship binds them to their parent entity.

Entities are also used to model high-level architectural elements, such as a file system. These high-level architectural elements are used when modeling storage relationships, and this use of entities to represent architectural elements is a departure from the typical database modeling use of E-R diagrams. Placing architectural elements and data model elements in the same diagram combines together two concerns that are usually separated. Architecture diagrams usually only contain architectural elements, and do not

address data modeling issues, while data models only contain data items, and do not address architectural issues. By combining them, storage control choices much clearer. For example, the defining difference between link server and hyperbase systems is control over storage, with hyperbase systems preferentially providing storage for works and link server systems delegating this storage to the filesystem instead. By making storage relationships explicit, the difference between open hyperbase and link server systems is more visible.

Entity-relationship models have entities composed of attributes. This has two drawbacks. One is that it privileges the entity, at the expense of the attribute, rather than treating abstractions uniformly. Since entities have an inclusion containment relationship with attributes, entities are treated as the primary data item, the thing that is described by the secondary items, the attributes. Second, it creates a special category for the aggregation relationship between entities and attributes, rather than treating it as just one point in a larger design space of containment. We prefer to treat entities and attributes the same, calling both entities, and then explicitly model the containment relationship between them. Hence, when creating data models for hypertext systems, data aggregation will be represented using inclusion containment relationships with the characteristics of single containment, single membership, and no ordering.

In the graphical representations of data models, a rectangle will be used to represent an entity defined by a hypertext system. An unboxed textual label represents other entities that exist outside the hypertext system, such as a file, or filesystem.

2.1.2 Relationships

There are three relationship types used when creating data models of hypertext systems: containment, inheritance, and storage. In graphical representations, an arrow-tipped line represents a relationship. Relationships are directional, and exist in both directions. So, for example, a container entity "contains" other entities, which are "contained by" the container.

The containment relationship is used to represent sets of entities. Containers have two main aspects, described below (and also presented in [34]).

Abstract properties of the container: Qualities of the container that are mathematic set properties, rather than properties of a specific computer representation, these being:

- *Containment*: For a given entity, the number of containers that can hold it. Choices are: (a) single containment, an entity belongs to just one containment set, or (b) multiple containment, an entity belongs to multiple containment sets,
- *Membership*: For a given container, the number of times it can contain a given entity. Choices are: (a) single membership, an entity can belong to a containment set only once, or (b) multiple membership, an entity can belong to a containment set multiple times, in which case the containment set is a bag, or multiset,
- *Ordering*: The persistent ordering of a container. Choices are: (a) ordered, the entities within the containment set have a fixed successive arrangement, or (b) unordered, the entities have no prescribed arrangement, (c) indexed, the arrangement is determined by a specification based on entity values or metadata, (d) grouped, subsets of members are ordered, but between subsets there is no ordering (e.g., {{a,b},{c,d,e}} or {{c,d,e},{a,b}}).

Containment type: How containment relationships are represented: (a) inclusion, or (b) referential (both described below).

Broadly, there are two ways to represent that a container contains a particular entity. The container can physically include the contained item, or it can use an identifier as a reference to its members. The former case is known as inclusion containment, the latter, referential. Whenever two entities have a containment relationship between them, this relationship can be represented using references, following the permutations of identifier storage: the identifier can be stored on the container, on the containee, or on both. Additionally, identifier storage can be delegated to a separate entity, a first-class relationship. However, since the first-class relationship is itself a container, the same permutations of identifier storage apply between the container and one endpoint of the relationship, and between the containee and the other endpoint. Typically the first-class relationship holds identifiers for both the container and containee.

Containment relationships have cardinality, depicted as numbers or the letters M and N (more than one, or many) on the relationship, expressing the number of entity instances that can exist at each end of the relationship. Note that the number at the container end of the containment relationship must agree with whether it is single containment or multiple containment. Since single containment indicates the entity can only be contained by a single collection, it must be represented by a "1", while multiple containment is represented by M or N, reflecting that the object can belong to multiple containers.

The inheritance, or "is-a", relationship is used *only* to avoid visual clutter due to the duplication of similar entities in the data model. Recapitulating the complete inheritance hierarchy is not a goal, since this is better accomplished by a separate diagram focused on inheritance relationships. Entities inherit all of the relationships of their parent, thus avoiding the need to duplicate all of these relationships on each child. Following the graphical convention given in [19], inheritance relationships are graphically represented using a thick double line.

The storage relationship represents that a specific architectural element provides physical storage for an entity. Storage can be viewed as a specialized form of inclusion containment relationship, where the containing entity is outside the set defined by a specific hypertext system. For example, in the Intermedia system, the filesystem provides storage for hypertext webs, and for documents [38]. It is useful to model this fact, since webs are stored separate from linked documents, and hence it is possible to have multiple webs over the same set of documents. Storage relationships are only used when the storage of entities is split among multiple architectural elements, as in a link server system, where objects are stored separate from the links between them. When only a single architectural element stores all entities, as in most hyperbase systems, storage relationships are omitted for clarity. The graphical depiction of a storage relationship is a thick solid line.

Figure 1 (below) provides the key for the graphical notation used in containment data model diagrams. Solid lines represent inclusion containment, and dotted lines represent referential containment. A single circle at the head of a relationship modifies

the relationship to indicate it is ordered, and a second circle indicates multiple membership. It is only in the data models of hypertext versioning systems that multiple membership and ordered referential containment are encountered.

2.2 Advanced Containment

The aspects of containment described in the previous section cover the majority of containment relationships, as is termed basic static containment. However, some systems provide advanced containment functionality, such as constraints on the containment relationships, and dynamic containment, which allows included objects to be determined by a computational process. This advanced containment capability is described below.

2.2.1 Containment Constraints

Some containers offer additional capabilities that extend the containment design space. These are:

Type of contained objects: Especially in systems that support a wide variety of object types, containers may limit, or explicitly state the type of objects that can be contained. For example, in the Hypermedia Version Control Framework [18], the association set is a container that can only contain associations, and in Aquanet [22], schema relations are containers that may restrict the type of contained objects. This issue is noted as the "Type" aspect of composite design in [13], p. 84. Containment data models represent this constraint by having containment arcs to all of the allowed entities.

Number of contained objects: Containers may have a fixed size, or an upper bound on their size. For example, Aquanet schema relations can have a fixed number of contained items, such as when modeling an argument relation (see Figure 2 of [22]), which has two slots for statements, and one slot for rationale. When only one kind of entity is contained, such as when a link contains only two anchors, the containment relation's cardinality can express the number of contained objects. But, if the constraint is of the sort, "two instances of the following three entities", then this must be expressed in a text description associated with the diagram.

Typing of the container: the previous two capabilities, specifying the number and type of contained objects, can be viewed as two kinds of constraints that would be given in the definition of a specific container type. Aquanet schemas essentially define new container types with each new relation, and DHM [11] provides several container subclasses, such as the GuidedTourComposite and TableTopComposite. Specific container types can express a wide range of structures, as noted in [13], p. 84-85. Compound documents can also be expressed using a container type that provides viewing and editing semantics for the contained objects that allows the container to behave like a single document, instead of a set of independent objects. Subtyping is represented using inheritance relationships in data model diagrams.

2.2.2 Dynamic Containment

For all containment types except for inclusion, a container can be viewed as a mapping from the set of all objects to the set of all containers. For single containment, this mapping is M:1, where M is the number of objects, while for multiple containment, this mapping is M:N, where N is the number of collections. With static containment, the set of members is explicitly listed. For dynamic containment, the mapping from objects to collections is generated by a function.

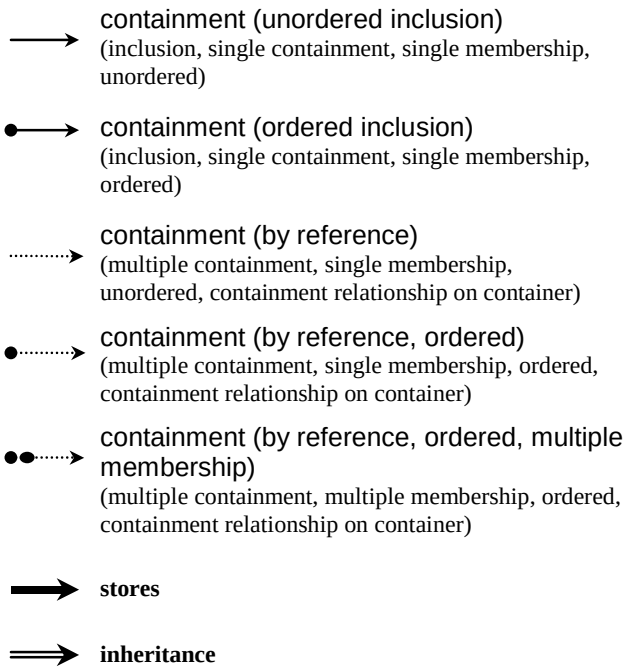

Figure 1 - Key to the graphical notation for relationships in containment data model diagrams.

Queries are by far the most common functions used to dynamically create containers. DHM [11], the Hypermedia Version Control Framework [18], and CoVer [14] are all examples of systems that support the population of containers from query results. There are two ways dynamic containment can be employed:

Query results specify the endpoint of one containment relationship: In this approach, each containment relationship can have a query associated with it, and this query typically is designed to return a single result, the endpoint. Thus the query determines a single element of the container. This approach is frequently employed to pick out a single revision from all the revisions of an object by scoping the query to just a single versioned object, and by selecting a query predicate that returns just a single revision. When scoped to a single revision history, the query predicate is termed a *revision selection rule*. The Hypermedia Version Control Framework supports arbitrary queries for endpoint selection [18].

Auld Leaky [32] provides another example of querying for endpoints, in this case, for the source of a link. The set of links that emanate from a specific node is determined dynamically by querying the link database for all links that have a matching context specification. Here "context" is a set of attribute-value pairs, representing the state of the reader, character, or plot during the process of reading a hypertext.

Query results specify the membership of a collection: Here the results of a query comprise the entire contents of the collection. In DHM [11], one system that supports this kind of containment, these containers are called virtual composites.

2.3 Relationship Abstraction Layers

At its most abstract, a container has an undifferentiated *contains* relationship between itself and its containees. The abstraction layer holding this undifferentiated contains relationship is termed the *abstract relationship layer*, since it provides an abstract depiction of the contains relationship, providing only the type of the relationships, omitting all other details concerning its specific properties. Entities in this layer are abstraction signifiers, indicating that they have distinct intellectual identity as abstractions, irrespective of whether their eventual concrete representation has independent identity. The abstract relationship layer is shown at the top of Figure 2.

Precisely specifying the characteristics of the relationships in the abstract relationship layer results in a more detailed depiction in the *explicit relationship layer*. Containment relationships at this layer have fully specified their containment, membership, and ordering properties, along with whether they are using inclusion or referential containment. Similar to the abstract relationship layer, entities in the explicit relationship layer are abstraction signifiers, and may not have distinct identity in the concrete representation layer. No refinement of entities occurs between the abstract and explicit relationship layers – only the relationships are refined. The explicit relationship layer is shown in the middle of Figure 2, which depicts two possible ways to refine the contains relationship in the abstract relationship layer into explicitly defined containment relationships.

In the *concrete representation layer*, abstract entities and relationships have been reified into specific data structures and chunks of state. Similarly, there are many possible computer data structures that can be used to represent a specific container as concrete data items [1]. Examples include arrays, linked lists, hashed lists, comma-separated text strings, and various types of trees, to name just a few. These data structures all support the operations of creating a set (or bag—all the following operations apply to bags too), inserting a member in a set, listing the members of a set, deleting a member from a set, and deleting a set. Ordered sets add position information to the insert operation, and additionally add an operation to order some members of the set. Other set operations are also possible, such as union, intersection, difference, etc., but are less frequently used by containers.

Repositories such as databases and filesystems can be used to realize the concrete representation. These systems themselves are complex, and often have several layers of abstraction within their implementation. Figure 2 shows one possible concrete representation, out of the universe of possible representations, for each of the examples in the explicit relationship layer. The inclusion containment example is reified as a file that internally has a linked list of data chunks, which are each a sequence of bytes. The referential containment example is represented using a container data item that holds within it a linked list of identifiers to contained data items. The internal structure of contained data items is unconstrained. Both the container and containee data items have identifiers.

3. HYPERTEXT SYSTEM DATA MODELS

Having fleshed out the details of containment data modeling, we now examine the fidelity and generality of this technique by modeling a broad set of systems spanning the categories of monolithic, link server, and (open) hyperbase, and including the

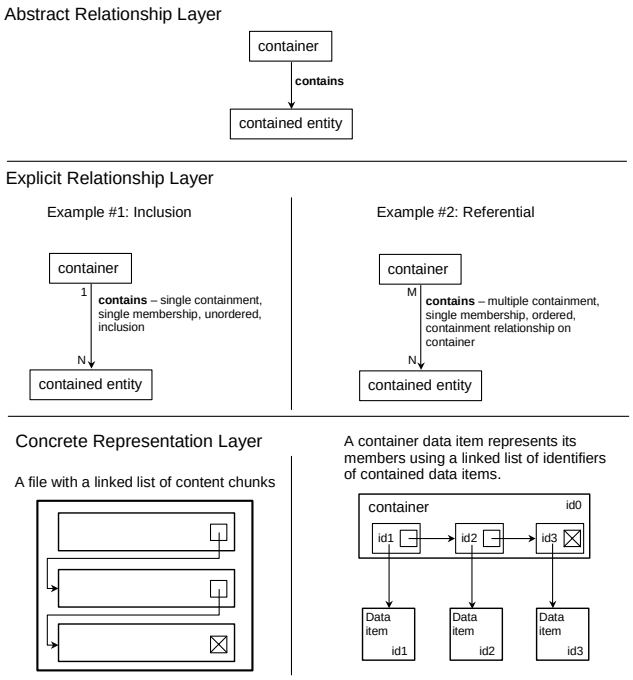

Figure 2 – A container at three different layers of abstraction. An abstract containment relation (abstract relationship layer) is refined into one example each of inclusion and referential containment. The explicit relationship layer fully details the containment relation, specifying containment, membership, ordering, and containment type. For the inclusion and referential examples, the concrete representation layer shows an example reification of the containers as persistent data items.

Dexter reference model and the WWW with WebDAV extensions. We use the Flag model's [24] assignment of systems to categories. Altogether, these figures comprise a significant survey of existing hypertext system data models.

3.1 Monolithic Hypertext Systems

A representative set of monolithic hypertext systems are shown in Figure 3, which models the NoteCards [31], KMS [2], Intermedia [38], HyperCard [4] and StorySpace 1 systems [5]. In this diagram, the modeled entities directly correspond to those of each system. So, while the notion of a NoteCards "notecard", a KMS "frame", and a HyperCard "card" are all similar (they are "card shark" systems), each system's data model uses the system-specific term for the card-like entity, instead of a more abstract term. Each system is modeled directly, without any additional mapping needed beyond assigning system abstractions to entities.

The KMS and HyperCard systems both have links that are embedded within a frame/card. These embedded links do not have distinct identifiers for each link, and hence their identity is not separable from their containing frame/card. Nevertheless, despite not having separate identity, these links do play a conceptually distinct role in the behavior of each system; they are essential to their "hypertext-ness". Thus, the links are modeled as distinct entities, ones that have an ordered inclusion containment relationship with their parent frame/card. Each link is modeled as a container, one that contains just a single member.

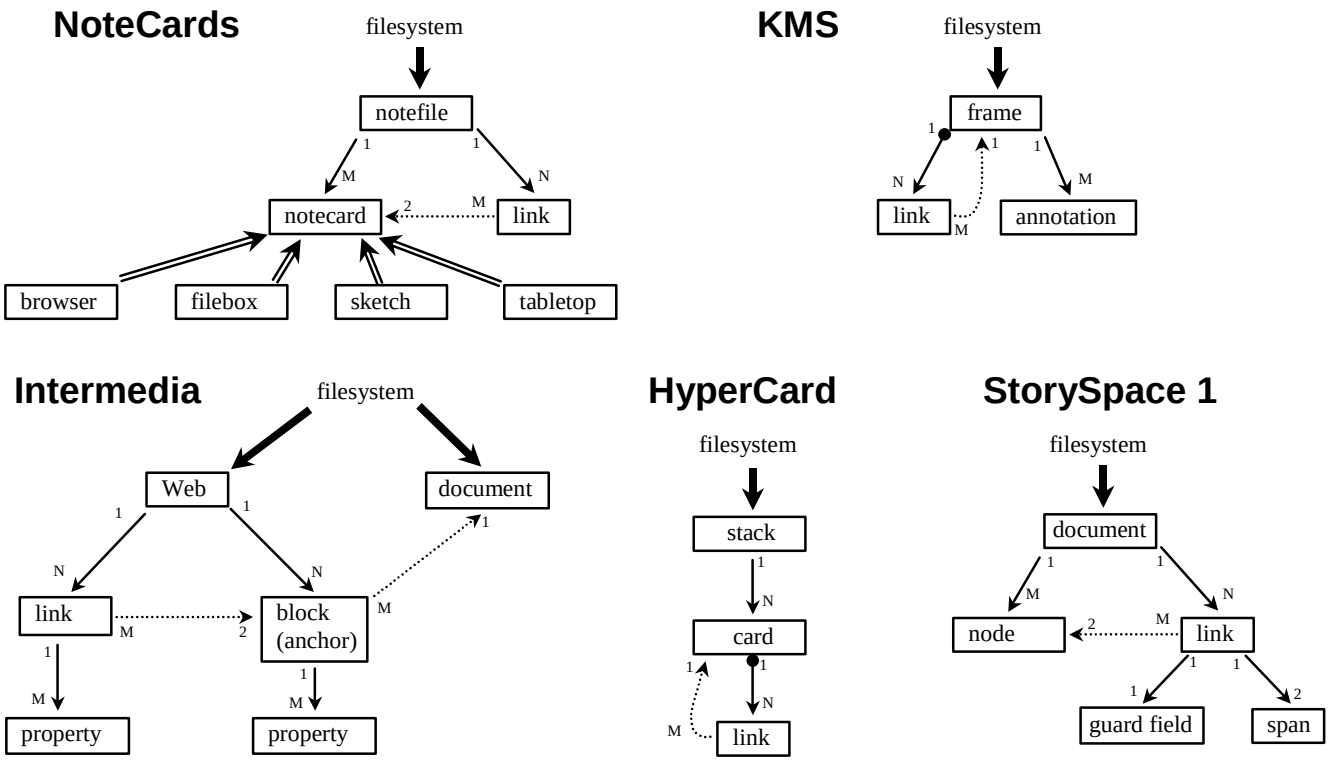

Figure 3 – Data models of selected monolithic hypertext systems: NoteCards [31], KMS [2], Intermedia [38], HyperCard [4], and StorySpace 1 [5].

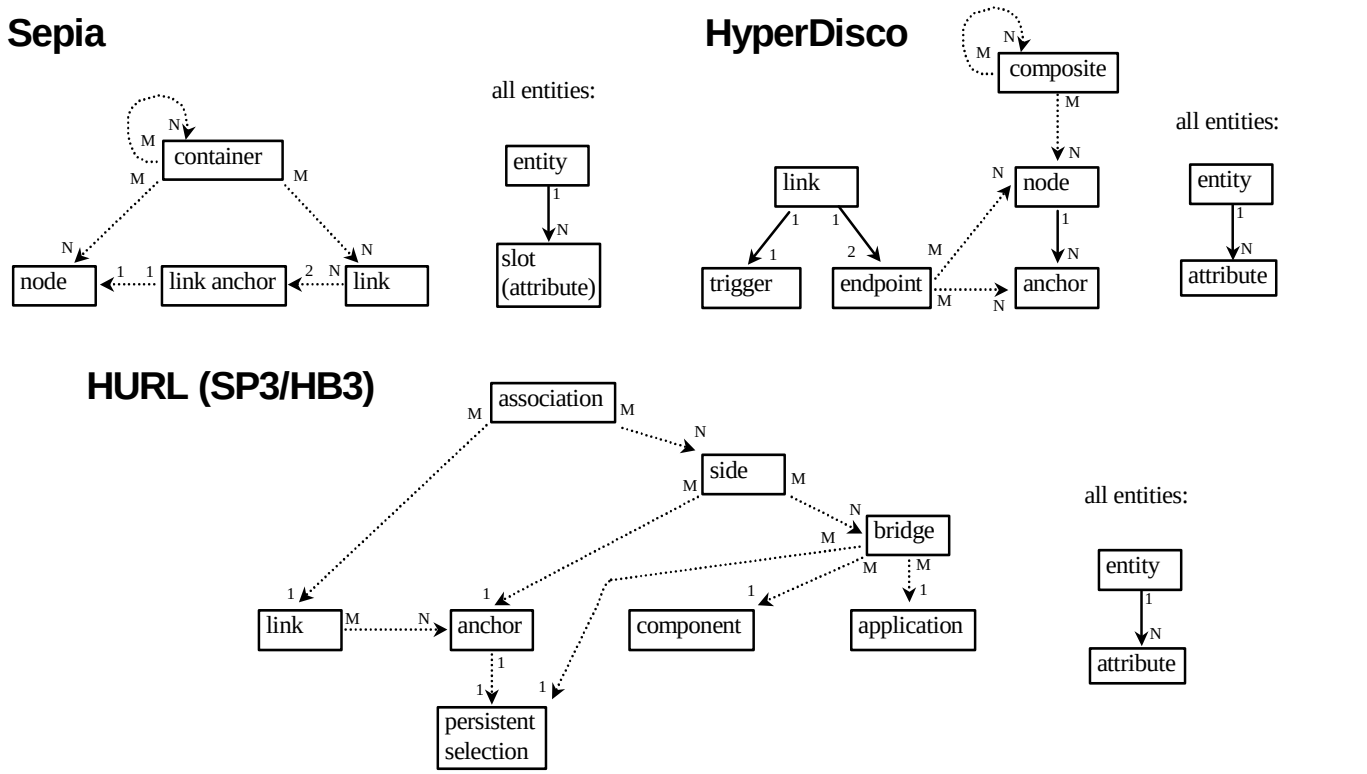

Figure 4 – Data models of selected hyperbase systems: Sepia [30], HyperDisco [37], and HURL (SP3/HB3) [18]. All entities are stored by the hypertext system, hence storage relationships are not explicitly shown.

The specific concrete representation of these link containers is a single identifier that specifies the link destination (the member of the link container), embedded within its parent frame/card.

The NoteCards model demonstrates the use of inheritance relationships to describe how browser, filebox, sketch, and tabletop cards are all specializations of the base notecard entity. Despite using inheritance relationships, this diagram does not give a complete inheritance hierarchy, instead focusing on only those cases where inheritance relationships reduce diagram clutter, and make the inter-entity relationships more clear. For example, it is also possible to model this system without inheritance relationships, but this would entail adding four more inclusion containment relationships between the notefile and all of the notecard types, and four additional referential containment arcs between the link and the notecard types. The diagram would be more cluttered, and make the NoteCards data model appear more complex.

The StorySpace 1 and NoteCards models show how containment data models can highlight similarities between system data models, since both models have a single file that inclusively contains nodes and links that otherwise act as first-class objects in the model. StorySpace can be viewed as a refinement of the NoteCards data model, adding the notion of guard fields and anchors (spans) to its links.

3.2 Hyperbase and Open Hyperbase Systems

Selected (open) hyperbase systems are shown in Figure 4, which gives containment data models of Sepia [30], HyperDisco [37], and HURL (SP3/HB3) [18]. Hyperform [36] is not modeled because it is a toolkit, with no normative data model of hypertext concepts. DeVise Hypermedia (DHM) [12] is not modeled, since a research goal of this system was to use the Dexter reference model as its internal data model, and Dexter is modeled separately, in Figure 6 (below). The containment data models of hypertext versioning systems, many of which are extended hyperbase systems, are discussed generally in previous work by the author [34], and are not repeated here.

In the hyperbase systems modeled, all hypertext entities contain a set of attributes, modeled as an inclusion containment relationship between object and attributes. Instead of adding significant additional clutter to the diagrams by showing every entity containing its set of attributes, a modeling convention is used that allows properties of all entities to be described in one place, just to the right of the main data model diagram. Object-oriented modeling would depict this as a base object, from which all other object types are inherited. This type of modeling could have been used as well, but at the cost of adding an additional inheritance arc for each hypertext system entity. This would add clutter to the diagram, defeating the purpose of factoring out the commonality of each entity containing a set of attributes.

The HURL data model (also the data model for SP3/HB3 [20]) clearly stands out as unique among all system data models. While the “bridge” is somewhat similar to Chimera’s “view” concept, the notion of “side” is unique, as is its definition of association. Having three separate paths from an association to a persistent selection is also unique. Containment data modeling, by concisely depicting the HURL data model in context with other systems, makes it possible to quickly identify the unique aspects of this data model. It also suggests additional avenues of research, such as examining how other system data models would behave if entities like sides and bridges were introduced.

3.3 Link Server Systems

Link server systems are shown in Figure 5 (below), which depicts the data models of Sun’s Link Service [25], Microcosm [9], Multicard [27], and Chimera [3]. Since the goal of link server systems is to provide third-party links among a set of documents, these systems often have an entity that represents, inside the link server system, a particular external object. This external object signifier is modeled as a container that contains a single member, the external object. The containment type is always referential, since the concrete representation of the container is an identifier of the external object (often a filename). The same modeling step is used for Chimera’s viewer, which is a single-member container referentially containing an application program.

Link servers highlight the utility of the storage relationship. By explicitly adding the filesystem as an entity to the model, it is possible to represent which entities are under the control of the link server system, and which are in its external environment. The storage relationship shows that the externally linked objects are stored by the local filesystem (and hence outside the control of the link server system). Entities inside the link server system are inside a link database, which itself is stored by the filesystem. Inside the link database, the link server system controls all entities.

The data model diagram allows cross-system comparisons. The data models of Sun’s Link Server and Microcosm are less complex than those of Chimera and Multicard. Chimera and Multicard both have additional grouping mechanisms, due to the presence of a self-containing container in their data models, namely the Chimera hyperweb, and the Multicard group. Chimera’s hyperweb only contains anchors and links (in addition to other hyperwebs), and hence is directly comparable to Intermedia’s “Web”, whereas the Multicard group can only contain nodes (in addition to other groups), and hence is similar to Dexter and HyperDisco composites, and WebDAV collections. Modeling a broad range of hypertext systems using containment modeling makes these kinds of cross-system comparisons much less difficult.

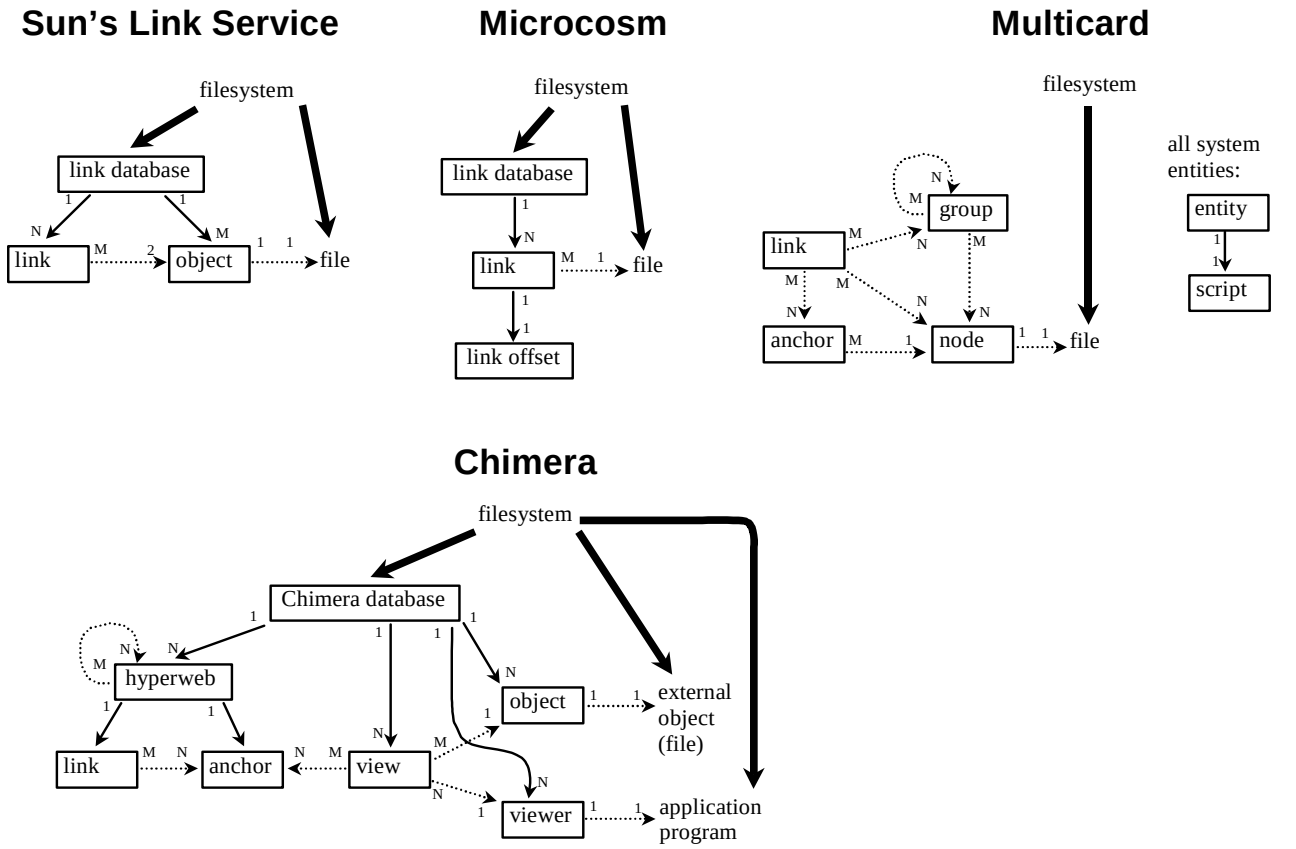

Figure 5 – Data models of selected link server systems: Sun's Link Service [25], Microcosm [9], Multicard [27], and Chimera [3]. All Multicard entities are stored within the Multicard persistent storage platform (not shown for clarity).

3.4 Dexter Reference Model

Figure 6 shows the Dexter hypertext reference model [16,17]. This diagram highlights several advantages to containment data modeling. First, the diagram collects in one place a number of facts that are spread throughout the text in [17] and the formal model in [16], allowing far more rapid understanding of much of the Dexter model than from the labor-intensive study of the primary sources (an observation that applies to other system data models as well).

The fact that Dexter can be modeled using containment data modeling means the modeling mechanism has good uniformity, able to represent both hypertext systems and cross-system reference models. It also paints a picture of the Dexter reference model as one member of a family of hypertext data models, embodying its own specific design choices and tradeoffs. Hence, containment modeling allows the Dexter model to be directly compared with its peers, the data models of other hypertext systems. Consider Dexter links, which contain two endpoints, where each endpoint contains a document and an anchor, which must be inclusively contained by the document. These links are very similar to those in HyperDisco, but are otherwise quite different from other hypertext systems, where a link typically contains either an anchor, or a document, but not both. Dexter composites are similar to HyperDisco composites, WebDAV collections, and MultiCard groups which all only contain documents, in addition to self-containment.

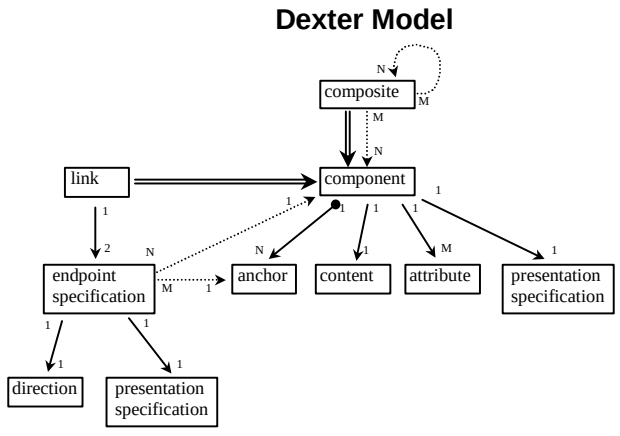

Figure 6 – Data model of the Dexter hypertext reference model [16,17].

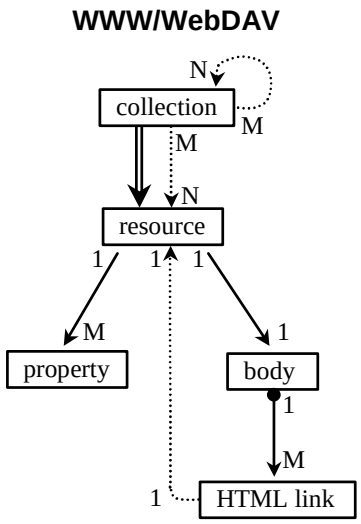

Figure 7 – Data model of the Web with WebDAV [33] extensions. HTML links also model other document types that have embedded links.

3.5 WWW/WebDAV

Figure 7 shows the containment data model of the WWW, with WebDAV extensions for remote collaborative authoring [35]. WebDAV adds to the base Web data model entities for containment, and for properties (attributes). As is the case for the Dexter model, the WebDAV containment model allows the Web's data model to be compared, in a uniform way, with other hypertext data models. While the Web is often treated as a special case among hypertext systems, due to its ubiquity and impact, from a data model perspective the Web is not unique. It can be grouped among the data models of hyperbase systems without seeming too far out of place.

The Web has a data model for links that is directly comparable to the KMS and HyperCard models, since all three have the pattern of a document inclusively containing links, which in turn referentially contain a single document. The primary different between the links is the form of identifier used in the concrete representation, with the Web using the Internet scale URL. Indeed, the similarity between KMS and the base Web data model (Figure 7 without containers or properties) is striking, begging the question of how history may have been different had KMS developed a protocol like HTTP, rather than depending on a network file system, and its scalability limitations.

4. CONCLUSION

Containment data modeling provides a modeling mechanism capable of uniformly representing the data models of a wide range of existing hypertext systems. Containment data modeling has been validated by presenting the models of 14 existing hypertext systems and reference models, the broadest survey to date of these models.

The uniformity of containment modeling is highlighted by the ability to decompose and model both Dexter and the WWW in the same way as other systems, allowing them to be compared with each other and with other systems using a consistent model. The understanding of Dexter and the WWW that emerges from this process shows them to be non-distinguished peers with other hypertext systems, carrying their own design choices and tradeoffs, but otherwise with no special distinction to their data models.

Containment data modeling provides a new technique useful in comparing hypertext system data models. By concisely representing system data models, and then grouping thems together, it is possible to quickly see similarities and differences among systems, including patterns for handling composites, anchors, and links.

Since containment data modeling focuses on modeling the static aspects of system data models, it is complementary to architecture-focused models, such as Flag [24], and formal models of system semantics, such as the FOHM model [23], or Trellis' Petri-nets [10]. While containment data models are a powerful and useful modeling technique, they alone do not give a complete picture of a hypertext system, and should be used in conjunction with other modeling techniques, providing multiple views of distinct aspects of each system.

Containment data models show significant promise for modeling systems in other domains, such as Software Configuration Management and Document Management. In our future work, we look forward to extending this technique to these additional classes of systems, enabling us to perform substantive cross-domain comparison of information management systems.

REFERENCES

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data Structures and Algorithms*. Reading, Mass.: Addison-Wesley, 1983.

[2] 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 (1988), pp. 820-835.

[3] K. M. Anderson, "Data Scalability in Open Hypermedia Systems," *Proc. Hypertext'99*, Darmstadt, Germany, February 21-25, 1999, pp. 27-36.

[4] Apple Computer, *HyperCard Script Language Guide*. Reading, MA: Addison-Wesley, 1988.

[5] M. Bernstein, "StorySpace 1," *Proc. Hypertext 2002, The Thirteenth ACM Conference on Hypertext and Hypermedia*, College Park, MD, June 11-15, 2002.

[6] G. Booch, *Object Oriented Design with Applications*. Redwood City: Benjamin/Cummings, 1991.

[7] P. Chen, "The Entity-Relationship Model: Toward a Unified View of Data," *ACM Trans. on Database Systems*, vol. 1, no. 1 (1976), pp. 9-36.

[8] N. M. Delisle and M. D. Schwartz, "Contexts-A Partitioning Concept for Hypertext," *ACM Transactions on Office Information Systems*, vol. 5, no. 2 (1987), pp. 168-186.

[9] A. M. Fountain, W. Hall, I. Heath, and H. C. Davis, "Microcosm: An Open Model for Hypermedia with Dynamic Linking," *Proc. First European Conference on Hypertext (ECHT'90)*, Versailles, France, Nov. 27-30, 1990, pp. 298-311.

[10] R. Furuta and P. D. Stotts, "Programmable Browsing Semantics in Trellis," *Proc. Hypertext'89*, Pittsburgh, PA, Nov. 5-8, 1989, pp. 27-42.

[11] K. Grønbæk, "Composites in a Dexter-Based Hypermedia Framework," *Proc. 1994 European Conference on Hypermedia Technology (ECHT'94)*, Edinburgh, Scotland, Sept. 18-23, 1994, pp. 59-69.

[12] K. Grønbæk and R. H. Trigg, "Design Issues for a Dexter-Based Hypermedia System," *Communications of the ACM*, vol. 37, no. 2 (1994), pp. 40-49.

[13] K. Grønbæk and R. H. Trigg, *From Web to Workplace: Designing Open Hypermedia Systems*. Cambridge, MA: MIT Press, 1999.

[14] A. Haake, "Under CoVer: The Implementation of a Contextual Version Server for Hypertext Applications," *Proc. Sixth ACM Conference on Hypertext (ECHT'94)*, Edinburgh, Scotland, Sept. 18-23, 1994, pp. 81-93.

[15] A. Haake and D. Hicks, "VerSE: Towards Hypertext Versioning Styles," *Proc. Seventh ACM Conference on Hypertext (Hypertext '96)*, Washington, DC, March 16-20, 1996, pp. 224-234.

[16] F. Halasz and M. Schwartz, "The Dexter Hypertext Reference Model," *Proc. NIST Hypertext Standardization Workshop*, Gaithersburgh, MD, Jan 16-18, 1990, pp. 95-133.

[17] F. Halasz and M. Schwartz, "The Dexter Hypertext Reference Model," *Communications of the ACM*, vol. 37, no. 2 (1994), pp. 30-39.

[18] D. L. Hicks, J. J. Leggett, P. J. Nürnberg, and J. L. Schnase, "A Hypermedia Version Control Framework," *ACM Transactions on Information Systems*, vol. 16, no. 2 (1998), pp. 127-160.

[19] R. Hull and R. King, "Semantic Database Modeling: Survey, Applications, and Research Issues," *ACM Computing Surveys*, vol. 19, no. 3 (1987), pp. 201-260.

[20] J. J. Leggett and J. L. Schnase, "Viewing Dexter with Open Eyes," *Communications of the ACM*, vol. 37, no. 2 (1994), pp. 76-86.

[21] E. Levinson, "The MIME Multipart/Related Content-type,". Internet Request for Comments (RFC) 2387, August 1998.

[22] C. C. Marshall, F. G. Halasz, R. A. Rogers, and W. C. Janssen, Jr., "Aquanet: a hypertext tool to hold your knowledge in place," *Proc. Third ACM Conference on Hypertext (Hypertext'91)*, San Antonio, Texas, Dec. 15-18, 1991, pp. 261-275.

[23] D. E. Millard, L. Moreau, and H. C. Davis, "FOHM: A Fundamental Open Hypertext Model for Investigating Interoperability between Hypertext Domains," *Proc. Hypertext 2000*, San Antonio, Texas, May 30-June 4, 2000, pp. 93-102.

[24] K. Østerbye and U. K. Wiil, "The Flag Taxonomy of Open Hypermedia Systems," *Proc. Seventh ACM Conference on Hypertext (Hypertext'96)*, Washington, DC, March 16-20, 1996, pp. 129-139.

[25] A. Pearl, "Sun's Link Service: A Protocol for Open Linking," *Proc. Hypertext'89*, Pittsburgh, PA, Nov. 5-8, 1989, pp. 137-146.

[26] J. Peckham and F. Maryanski, "Semantic Data Models," *ACM Computing Surveys*, vol. 20, no. 3 (1988), pp. 153-189.

[27] A. Rizk and L. Sauter, "Multicard: An open hypermedia System," *Proc. Fourth ACM Conference on Hypertext (ECHT'92)*, Milano, Italy, Nov. 30-Dec. 4, 1992, pp. 4-10.

[28] J. L. Schnase, J. J. Leggett, D. L. Hicks, and R. L. Szabo, "Semantic Data Modeling of Hypermedia Associations," *ACM Trans. on Information Systems*, vol. 11, no. 1 (1993), pp. 27-50.

[29] L. F. G. Soares, G. L. d. S. Filho, R. F. Rodrigues, and D. Muchaluat, "Versioning Support in the HyperProp System," *Multimedia Tools and Applications*, vol. 8, no. 3 (1999), pp. 325-339.

[30] N. Streitz, "SEPIA: A Cooperative Hypermedia Authoring Environment," *Proc. Fourth ACM Conference on Hypertext (ECHT'92)*, Milano, Italy, Nov. 30-Dec. 4, 1992, pp. 11-22.

[31] R. Trigg, L. Suchman, and F. Halasz, "Supporting Collaboration in NoteCards," *Proc. Computer Supported Cooperative Work (CSCW'86)*, Austin, Texas, Dec. 3-5, 1986, pp. 147-153.

[32] M. J. Weal, D. E. Millard, D. T. Michaelides, and D. C. D. Roure, "Building Narrative Structures Using Context Based Linking," *Proc. Hypertext 2001, The Twelfth ACM Conference on Hypertext and Hypermedia*, Århus, Denmark, August 14-18, 2001, pp. 37-38.

[33] E. J. Whitehead, Jr., "Goals for a Configuration Management Network Protocol," *Proc. 9th Int'l Symposium on System Configuration Management (SCM-9)*, Toulouse, France, Sept. 5-7, 1999, pp. 186-203.

[34] E. J. Whitehead, Jr., "Design Spaces for Link and Structure Versioning," *Proc. Hypertext 2001, The Twelfth ACM Conference on Hypertext and Hypermedia*, Århus, Denmark, August 14-18, 2001, pp. 195-205.

[35] E. J. Whitehead, Jr. and Y. Y. Goland, "WebDAV: A Network Protocol for Remote Collaborative Authoring on the Web," *Proc. Sixth European Conference on Computer Supported Cooperative Work*, Copenhagen, Denmark, Sept. 12-16, 1999, pp. 291-310.

[36] U. K. Wiil and J. J. Leggett, "Hyperform: Using Extensibility to Develop Dynamic, Open and Distributed Hypertext Systems," *Proc. Fourth ACM Conference on Hypertext (ECHT'92)*, Milano, Italy, Nov. 30-Dec. 4, 1992, pp. 251-261.

[37] U. K. Wiil and J. J. Leggett, "The HyperDisco Approach to Open Hypermedia Systems," *Proc. Seventh ACM Conference on Hypertext (Hypertext '96)*, Washington, DC, March 16-20, 1996, pp. 140-148.

[38] N. Yankelovich, B. J. Haan, N. K. Meyrowitz, and S. M. Drucker, "Intermedia: The Concept and the Construction of a Seamless Information Environment," *IEEE Computer*, vol. 21, no. 1 (1988), pp. 81-96.