A component-based open hypermedia approach to integrating structure services

Peter J. Nürnberg, Kaj Grønbæk, Dirk Bucka-Lassen, Claus Aagaard Pedersen, Olav Reinert
Department of Computer Science, Aarhus University
Aabogade 34A, DK-8200 Aarhus N, Denmark
{pnuern, kgronbak, dibuck, aagaard, oreinert}@daimi.au.dk

Abstract. In this paper, we consider the issue of integrating different structure services within a component-based open hypermedia system. We do so by considering the task of collaborative editing, which calls for a variety of different structures traditionally supplied by different structure services. We discuss the nature of collaborative editing and how it can be supported by a combination of spatial and navigational hypermedia services. We then present a component-based open hypermedia system architecture and describe various methods of integrating different structure services provided within such an architecture. We show the advantages of integration within a component-based framework over other means of integration, highlighting some of the main advantages of the component-based approach to open hypermedia system design and implementation.

Keywords: open hypermedia system (OHS), component-based open hypermedia system (CB-OHS), structure services, structural computing, spatial hypermedia, navigational hypermedia

1 Introduction

Over the past several decades, different hypermedia systems have been proposed to address a wide variety of problem domains. From traditional "navigational" hypermedia systems that support associative storage and recall [Bush 1945] such as the World Wide Web [Berners-Lee et al. 1994], to spatial hypermedia systems to support information analysis [Marshall and Shipman 1995]; from issue-based hypermedia systems to support argumentation and design rationale capture [McCall et al. 1990] to taxonomic hypermedia systems to support classification [Parunak 1991], hypermedia has been brought to bear on a wide variety of problems.

One characteristic of such different systems however, is that it has traditionally been difficult or impossible to use the structures created in one such system in another. Intra-domain interoperability and sharing of structure have long been recognized as important issues [Østerbye and Wiil 1996], and is currently being address by the OHSWG [1997], but inter-domain interoperability is essentially unexplored and even largely unrecognized as an issue. Although some initial attempts at using structural abstractions tailored for one domain in another domain have been made [Marshall and Shipman 1997], such integrations have been idiosyncratic and difficult to generalize.

Unfortunately, there are many cases in which precisely this kind of integration could be very useful for supporting complex work tasks. One such case is the domain of collaborative editing. The collaborative editing task involves generation and collection of content, organization of this content into a compilation, and delivery of this compilation. As we discuss in more detail below, the initial organization work required for building the compilation is well supported by spatial hypermedia systems, while the delivery of such compilations is well supported by navigational hypermedia systems. The spatial organization (structure) built up in the initial phase of the work should be able to be used again in the delivery of the final compilation form, requiring that structures built and managed by one service be used by another service designed for another problem domain.

There are several possible approaches to reusing structures built in one service in another. In this paper, we describe integration within a component-based open hypermedia system (CB-OHS) [Nürnberg et al. 1997]. A CB-OHS is much like a traditional OHS, but with an open middleware (structure service) layer, allowing many different types of hypermedia servers to exist side-by-side. Because these different structure services exist within a common multi-layer framework, there are many possibilities for integrating them. Below, we describe several of these, and discuss various tradeoffs. We also show the potential advantages of integrations based on a common framework to other approaches, such as retrofitting or translating structural abstractions built in monolithic or non-interoperating systems.

The remainder of this paper is structured as follows. In Section 2, we examine the domain of collaborative editing more thoroughly, and show how spatial and navigational hypermedia in conjunction can provide useful support for this task. We also present a scenario that illustrates users engaged in a collaborative editing task, using CB-OHS tools. In Section 3, we describe the conceptual architecture and prototypic implementation of a CB-OHS that supports collaborative editing tasks like the one illustrated by the scenario. In Section 4, we discuss a number of options for integrating different structure services within a CB-OHS framework. In Section 5, we review work related to integration of various hypermedia services and to hypermedia support of domains similar to collaborative editing. Finally, we present directions for future work and our conclusions.

2 Collaborative editing

This section describes collaborative editing, which is similar to but different from collaborative authoring. We claim that integrated structure services within a CB-OHS are highly desirable for supporting tasks in this domain. Firstly, we describe the domain of collaborative editing more fully. Secondly, we discuss how hypermedia abstractions can be applied to work in this domain. Finally, we provide a scenario that illustrates how people might use a CB-OHS to carry out collaborative editing.

2.1 Domain description

As stated above, collaborative editing is similar to collaborative authoring, which has long been a domain for which hypermedia systems have been developed. NoteCards [Halasz et al. 1987] and SEPIA [Streitz et al. 1992], for example, have been applied for such tasks. These systems provided support for the actual writing process as well as tools for brainstorming, outlining and planning of documents.

Collaborative editing tasks can be seen as a variant of collaborative authoring tasks, since less emphasis is placed on the actual generation of content, and conversely, more emphasis is placed on integration of already existing sources. Of course, original content generation must be supported. However, collaborative editing tasks concern both original content integration and secondary content integration, which differentiates this domain from collaborative authoring. These tasks also focus heavily on creation and organization of the final product that contains this content.

In the collaborative editing domain, there are three main character roles. *Content locators* locate secondary content. *Content generators* generate original content and/or collect secondary content. Collectively, content locators and content generators may be called *content managers*. *Editors* organize content (either original or secondary) into compilations. Characters may assume multiple roles. There may be any non-zero number of characters fulfilling content manager and editor roles.

Location of secondary content requires facilities both to search content bases and for visualizing and manipulating search results. Generation of content requires all of the support present in systems that address the collaborative authoring domain. Organization of compilations is an iterative and evolving process, and may point out the need for new content.

2.2 Application of hypermedia to collaborative editing

A hypermedia-oriented (or structural computing) approach to analyzing the collaborative editing domain requires us to consider the nature of both the *data* and *structural* abstractions used by workers engaged in this type of work. We would like to make as few assumptions about the data (content) as possible, to allow any analysis to be as generally applicable as possible. Therefore, we take content to include but not be limited to: existing Web resources (URLs), existing non-web resources in WordPerfect, Word, Excel or other formats (local file system URLs) and placeholders for new documents to be located or generated. We also make no particular assumptions here about how content is generated or found, although these are important and relevant issues in general.

Below, we focus on the creation, evolution, and delivery of the compilation. We first examine the usefulness of spatial hypermedia to aid users in organizing compilations. We then look at how navigational hypermedia can be used to support delivery of compilations. We end with a brief note about the desirability of openness in such an environment.

2.2.1 Organizing a compilation

Organization of compilations requires flexible and evolving structures to reflect the iterative nature of this task. These structures must help alleviate the problems associated with premature formalization [Halasz 1988]. Spatial hypermedia structures are well-suited for this kind of evolving organization task.

Spatial hypermedia can be thought of as using a note card metaphor for organizing information. For example, consider the task of organizing a journal paper. One might begin by generating many note cards, each containing a description of some part of the paper. During the generation process, one might start to organize these cards on a table, perhaps grouping cards into piles, lists, etc. Placement on the table space might also be significant. For example, cards might be roughly sorted by priority, with more important cards being placed toward the upper left corner, and less important cards toward the lower right corner. Cards can be easily rearranged as one's understanding of the paper organization evolves. At no time is one required to declare formally an overall structure to the cards on the table. At some point, one might deem the organization task to be complete enough to allow the paper generation task to begin.

Spatial hypermedia systems support this note card metaphor of organization by allowing "cards" to be generated and placed on a "table" (space). Cards may be tailored by changing their size, shape, color, or other visual characteristics. Cards may contain content or point to external content. Additionally, some systems allow cards to be "opened" to reveal another space that also may contain many different cards. Some systems provide so-called "spatial parser" that might recognize certain structures (such as piles, lists, etc.) and allow users to manipulate these structures as whole units (e.g., by allowing all the cards in a pile to be moved at the same time). Some systems allow repeated structures to be "formalized" (i.e., assigned a name, a description of a semantic role, etc.), which might allow the system to provide more functionality with respect to such formalized structures (e.g., search for other instances, etc.).

2.2.2 Delivery of a compilation

Delivery of material, whether compilation or otherwise, requires that consumers be able to switch their focus between the structures associating pieces of content and the content itself. So-called navigational hypermedia structures are by far the most commonly used for delivery of information.

By navigational hypermedia, we mean those kinds of structures that are the most familiar hypermedia abstractions; certainly, this includes nodes and links, but also perhaps anchors, endpoints, contexts, etc. The basic notion behind navigational hypermedia is to enable associative storage and recall of information. That is, in addition to allowing more traditional mechanical access to information through such well-known devices as full-text indexes or categorizations (e.g., alphabetization, Library of Congress classification, etc.), users are allowed to designate and then follow idiosyncratic associations or links between data items. In this model, data is "chunked" into pieces that are then represented by nodes. Anchors designate parts of nodes (or other objects). Endpoints associate an anchor with a link. Links may contain an arbitrary number of endpoints. Contexts group other abstractions. Opening or closing a context allows the abstractions in it to be seen by or hidden from end users, respectively.

Navigational hypermedia systems support association of data (wrapped by nodes) through linking. Different systems may support different semantic constraints (e.g., enforcing directionality of links) or may not include certain abstractions (e.g., binding anchors directly to links, thus removing the need for endpoints). Other systems may provide additional abstractions such as guided tours [Trigg 1998].

2.2.3 Openness

Most collaborative authoring systems (and, in fact, until recently, most hypermedia systems) including those mentioned above, were monolithic and therefore required all of the work to take place with a closed set of tools. In modern computing environments, users typically apply many different tools to accomplish their tasks. Thus, openness is a highly desirable characteristic in computing environments.

If a particular service is delivered in an open way, we mean that it is provided to clients orthogonally to other services. For example, if spatial hypermedia services are provided to clients, but in order to use these services all data generated by these clients must be stored within the spatial hypermedia service, this service is provided in a closed way, since it is not orthogonal to the persistent data storage service. Conversely, if clients are free to use these spatial hypermedia services without being forced to use any other service, it is then an open service.

2.3 Scenario

We now illustrate the support of collaborative editing within a CB-OHS environment in the following scenario inspired from newsletter production tasks in one of the user organizations involved in the Coconut project [CIT 1999] at Aarhus University.

The scene is an advisory service organization, in which a group of advisors (content managers and editors) produce advisory material for colleagues via the Internet and their organizational intranet. The advisors are experts in different subject areas; they are responsible for keeping up to date with applicable laws, research and other activities within their subject area. The advisors work in different branches of the organization and are connected to local file and Web servers constituting their daily workspace. The material they produce is issued as a weekly newsletter that takes the form of a document collection distributed across the local Web servers throughout the organization.

One advisor is appointed as the editor for the newsletter series that is issued on the Web servers. The editor is responsible for encouraging contributions, collecting advisors' proposals for contributions, commenting and editing proposals, planning what should appear in different issues of the newsletter, and ensuring that the contributors complete their tasks. With CB-OHS empowered tools, this work may be accomplished as described below.

2.3.1 Planning a Web-based newsletter with a CB-OHS

The editor, Edgar, starts brainstorming with his colleagues about this week's newsletter. Each user starts up a spatial hypermedia client running as an applet in a Web browser. Each of these clients is connected to a spatial hypermedia server. They agree to start up their clients in tightly coupled collaboration mode. They may also be using secondary communication tools for informal communication and coordination tasks (e.g., MBONE conferencing tools).

Edgar has created a new space for this week's newsletter. The advisors place cards in this space for a number of different documents to go into the compilation. They then collaboratively sort the cards into different piles in different locations with the spatial client. The spatial server is constantly recalculating the implicit structures in the space using an incremental spatial parser, and periodically saves persistent copies of these structures to the store. At some point, Edgar suggests that they divide the responsibilities among themselves according to the piles that are in the space. In this case, five collections (composites) of cards are present when the division of responsibilities occurs. Jens takes responsibility for collection 1 (C1), Susanne takes responsibility for C2 and C3, and Edgar, the editor, takes responsibility for C4 and C5. These collections are assigned titles, and will later appear as sections in the newsletter. That is, these piles (spatial structures) will be reused in the final compilation form as sections (navigational structures).

2.3.2 Writing and structuring newsletter sections (collections) with CB-OHS

The advisors now take responsibility for their document collections, and both create new resources (e.g., HTML documents) on their local servers and find existing resources that may either be local or remote. They associate these resources with cards in the newsletter space. Occasionally, this process points out the need for cards to be added to, removed from, or moved in the space. Advisors can move between content management and editorial tasks.

Some of the advisors subscribe to notifications on changes to the cards in some of their colleagues' collections, because they need to know when the content associated with those cards is ready for cross-referencing with links. These notifications take the form of decorations of the cards in the spatial client when the content associated with those cards is updated.

The individual advisors each use their favorite editing programs to generate content (Word, FrontPage, etc.). Each of the editing programs have been integrated with a navigational hypermedia server. Editing a document collection involves both writing of new text for the placeholder cards and linking of existing document sources. When editing new documents, the advisors may switch between creation of embedded links in HTML and creation of external bi-directional many-to-many links as well as global links[1] using the navigational client add-ons.

One of Susanne's collections (C2) is a collection of existing Web-documents produced by other companies and governmental offices. Susanne chooses to make C2 into a guided tour using the navigational client.

When the individual advisors have finished writing and organizing drafts for their collections, Edgar assigns some other colleagues to review the collections using the navigational client's annotation and

[1] Global link is a name for the Dexter-inspired generalization of Microcosm generic links [Davis et al. 1994] suggested by Grønbæk and Trigg [1996].

linking facilities. The reviewers' annotations and links are created within a separate context object that can easily be deleted when the editor and the author have agreed how to address the comments.

The final form of the Web newsletter has a title page with links to different sections (corresponding to the newsletter space with different collections). Each section has a header page with links to individual newsletter items (corresponding to the generated and/or located content associated with individual spatial cards).

3 Environment for service integration

In this section, we present the conceptual design and prototypic implementation of a CB-OHS. Firstly, we consider the infrastructure, considering the process architecture of entities in the CB-OHS environment, and interaction among processes. Secondly, we consider the different services available in a typical CB-OHS, dividing our discussion into backend and structure services. Although by their nature, CB-OHS's contain an open set of structure services, we describe spatial, navigational, session, and connection services, since they are especially useful for supporting the collaborative editing scenario presented above. Finally, we consider different kinds of applications that can be integrated into a CB-OHS. Figure 1 shows the conceptual architecture that forms the basis for our discussion.

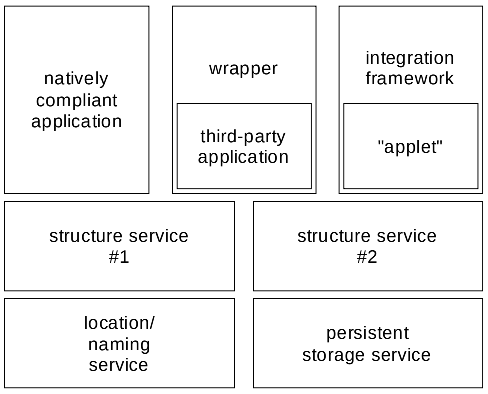


Figure 1. CB-OHS conceptual architecture.

3.1 Infrastructure

3.1.1 Process architecture

Each server in the environment may be conceived as consisting of several parts (see Figure 2). A virtual client represents a connection to a client entity, while a virtual server represents a connection to a server entity. Virtual clients are grouped into a virtual client pool. Likewise, virtual servers are grouped into a virtual server pool. Interaction between the virtual client pool and virtual server pool is mediated by the core. A session proxy consists of sets of virtual client pools, virtual server pools, and cores. Session proxies are created and managed by a connection manager. When client entities make a connection, they do so through the connection manager, which then creates a corresponding virtual client and places it in the appropriate virtual client pool in the appropriate (possibly newly created) session proxy.

The virtual client pool and virtual server pool provide a way to multiplex communication among various virtual clients and servers. That is, the core need only view communication as originating from a client pool and being directed toward a server pool, or vice versa. The logic for determining which virtual clients or servers should receive a particular message is transparent to the core.

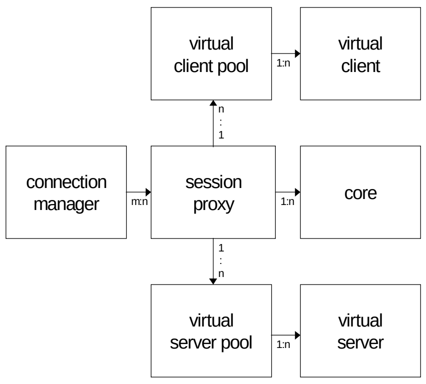


Figure 2. Process architecture.

The session proxy part of an entity represents one particular process' view of a session. An actual collaborative work session may consists of multiple session proxies, spread among multiple processes. This is described in more detail below in the section on session information services.

We have implemented the process architecture framework as a set of Java libraries. Construction of a new service entity involves subclassing abstract virtual client pool, core, and virtual server pool classes. Connection and session management, and communication via virtual clients and servers are all automatically provided by the framework. Communication can be effected by ASCII streams over TCP sockets or using Java's RMI facilities.

3.1.2 Interprocess interaction

Processes may directly interact in various ways. Logically, one process may be a client of a second, a server of a second, or a peer of a second. In practice, all interactions are manifested as peer-to-peer, since all interactions are realized as either request-response pairs or notifications-response pairs (in either of which, a response may be null). Logical peer-to-peer interactions are modeled as a pair of interactions, in which one process is the client of the other in one interaction, and the server in the other.

Processes may indirectly interact by being in the same collaborative work session. Each session is comprised of various state information, some of which may be shared (dependent upon the coupling mode of the session). As mentioned above, collaborative work sessions may span servers. Thus the session proxies described above in the context of process architectures are not sufficient to represent work sessions. Instead, we provide a session information service that allows for the creation, manipulation, and deletion of session records. A session record contains all relevant information about a work session, such as its name, joining policy, coupling mode, members, the tools (applications) and services being used, the documents and artifacts over which collaboration is taking place, etc.

3.2 Services

The following describes some of the services that might be found in a CB-OHS. We first describe backend services. These services are generally guaranteed to be available to all entities in the system. Then, we describe structure (middleware) services. Since this set is open, we choose four representative examples.

3.2.1 Backend services

This section describes backend CB-OHS services. Here, we look at persistent storage and naming/location services.

3.2.1.1 Persistent storage service

A CB-OHS provides its clients (structure servers and applications) general and flexible persistent structural abstractions. Although there are a number of similarities between a CB-OHS persistent store service and a traditional OHS store service, one essential difference is the greater generality of the abstractions in the former. Since an OHS persistent store is designed to support (in general) a closed set of structure servers, it can offer a high degree of tailorability at the expense of less flexibility. Conversely, because a CB-OHS has an open set of clients that require structure management support, its persistent store abstractions must be highly tailorable, and thus more generic. Some amount of tailoring work is therefore pushed onto the structure servers.

To combat this potential increased programming load, a CB-OHS storage service can exhibit a certain degree of extensibility with respect to tailoring basic abstractions. Specifically, structure servers should be able to view generic objects in the store as tailored objects, if the responsible store has been extended in the appropriate way. That is, a navigational hypermedia server should be able to view a generic object as, for example, a link object, if the store has the appropriate "link" extension. However, all store clients should be able to view this same link object generically, or even as other types of objects. It is important that all attributes of an object remain visible independent of the way in which it is viewed. This implies that any structure server can read and manipulate any object.

We have implemented a persistent storage facility for our CB-OHS in Java that communicates with an arbitrary JDBC accessible database management system (DBMS). We currently have integrated our storage service with Oracle and MS Access. Our store provides its clients a basic persistent storage abstraction called a *unit*. A unit consists only of an arbitrary set of characteristics and a set of metadata (e.g., creator, creation date, permissions, etc.). Each characteristic takes a set of values. Characteristics may be one of three basic types: attribute (for which values are arbitrary strings); behavior (for which values are identifiers of computations); and, relationship (for which values are identifiers of units). Our storage facility also extracts other abstractions, such as *account*, (*account) group*, *computation*, and *version set*. We have also implemented API extensions for each of the data abstractions in the OHP-Nav interface standard [OHSWG 1997], described in more detail below.

3.2.1.2 Naming and location service

In a highly distributed environment such as a CB-OHS, location (i.e., mapping names to server addresses) and naming (i.e., mapping names to objects) services are critical. We propose, independent of the actual implementation, that these services be tightly integrated, so that the server (structure server or storage server) that is "primarily responsible" for an object can be located given only the name (identifier) of an object. (The notion of "server of primary responsibility" is left ill-defined here, but can be thought of intuitively as the server that holds the primary persistent replicate.) This implies that any entity can locate the primary server of any object for which it is given the name.

We have implemented such an integrated location and naming service. The location service allows architectural entities to register service names along with a set of communication addresses and some other metadata about the service. Other entities can then query this service for the addresses at which various other services may be found. Clients may request a "naming domain" or identifier prefix from the naming service, to which they may append arbitrary strings to form identifiers. It is the responsibility of the requesting client that the assigned suffixes of these identifiers be made unique.

Our location and naming servers are organized into hierarchies. They propagate location information amongst themselves in a DNS-like manner and distribute naming authority by requesting naming domains from their parents. A root server contains (recursively) all information to resolve names in its domains or knowledge of which child location server can do so, and (recursively) holds authority for the shortest prefix identifier naming domain. There may be more than one root server, so global uniqueness of service and object names is not guaranteed, implying that cooperating sites must agree on a common root server.

3.2.2 Structure services

This section describes four representative structure (middleware) services. We look at navigational and spatial hypertext services like those described above in Section 2.2, and session and connection services, which help us manage general collaborative work.

3.2.2.1 Navigational hypermedia service

As our point of reference, we consider navigational services as defined by the OHSWG in the OHP-Nav draft standard [OHSWG 1997]. We briefly review the major concepts here. The OHSWG data model exported by compliant navigational servers consists of multi-headed, n-to-n, anchored links, nodes, and contexts. Anchors consist of an arbitrary location specifier and node identifier, while nodes consist of an arbitrary content specifier (URL). Nodes may wrap internally or externally stored data, or even other structural objects; likewise, location specifiers may designate a part of any type of content wrapped by a node. All objects may have an associated computation. In addition to services for basic manipulation of these objects, there are services to effect traversals of structure (from a set of "sources", which are location specifier/node identifier pairs, to a set of "destinations", which are also such pairs). Such traversals may activate computations associated with structural objects, thus allowing an open set of traversal semantics.

Navigational structure services represent the most venerable example of hypermedia functionality and are well-suited to supporting a variety of associative storage and recall tasks. There are a great number of projects in which navigational services much like those we describe above have been used to underlie delivery of materials to readers (e.g., NoteCards [Halasz et al. 1987] and SEPIA [Streitz et al. 1992]), especially for materials for which there is no required total reading order. It is an essentially navigational model of structure that underlies the by far most ubiquitous example of hypermedia, the WWW, which is used on an extremely wide scale for delivery of such reading material.

In addition to the abstractions defined in the OHP-Nav standard, our prototypic navigational server implementation serves a number of additional abstractions, including facilities that support guided tours. Guided tours are a type of highly structured composite that provides ordering information about its children. It is a kind of "super-structure" that itself structures traditional navigational abstractions such as links. Several projects have discussed the usefulness of guided tours as an augment to associative links [Trigg 1988; Zellweger 1989]. Our server is multi-user and multi-session, and implements the session semantics defined in the current OHP-Nav standard.

We have written wrappers for various well-known third-party applications, including MS Word, MS Internet Explorer, and Emacs. Because it is compliant to the applicable OHSWG standards, our navigational server works with clients and/or wrappers developed as parts of other OHSWG compliant implementation efforts (e.g., Solent, successor to Microcosm [Davis et al. 1994]).

3.2.2.2 Spatial hypermedia service

Spatial hypermedia systems have been used for a number of years to support information analysis tasks similar to those we describe above in the prewriting phase of collaborative editing [Marshall et al. 1991; Marshall and Shipman 1995]. We briefly review some of the major concepts here. The basic data model exported by a spatial server consists of a set of structured spaces. Each space (above, "card") consists of an arbitrary content specifier (URL), a description, a set of children that are also spaces, and a set of visual characteristics, such as color, size, and position within its parent space. Clients of the spatial server normally allow manipulation (creation, deletion, modification, movement, etc.) of icons that correspond to spaces on the server.

In addition to the structuring provided by spaces (i.e., composition of sibling spaces), there is also structuring based on visual characteristics of sibling spaces. Sibling spaces that are proximate and/or have similar visual characteristics may be grouped by a spatial parser that continually examines the children in each space for visual patterns. When a pattern amongst some number of sibling spaces is found, the spatial server creates a new "virtual" space (composite) that corresponds to this pattern. Virtual spaces are equivalent to normal spaces at the server, but are not rendered at the client. Virtual spaces represent implicit structures in the realized spaces, and can be used for the basis of such structure manipulations as hierarchic click selection [Marshall et al. 1994]. Patterns may also be explicitly recognized and thereby "realized" (i.e., be made non-virtual), much like the composite recognition in VIKI [Marshall and Shipman 1995].

In addition to the features of other spatial hypermedia systems such as Aquanet and VIKI, our prototypic spatial server implementation (CAOS, see [Bucka-Lassen et al. 1998]) uses an *incremental* parser that recalculates the set of virtual spaces upon client modification of realized spaces. Clients communicate with the server via a public protocol. Thus, new clients may be added to the system at any time, and can take advantage of both spatial abstraction persistence and the incremental parser. We have also built a custom made spatial hypermedia client that provides similar functionality to other spatial hypermedia systems mentioned above. Our server is multi-user and multi-session, and allows several different coupling modes, ranging from tightly coupled WYSIWIG to loosely coupled major edit notification.

3.2.2.3 Session service

Our session service provides clients with a way to share session information. The main abstraction is a session state, which contains a set of first-class fields and a set of arbitrary attribute/value set pairs. Some subset of this session state is designated as the session record. A session record contains the "public" information about a session, such as its name, its members, the servers participating in the session (i.e., the servers for which there are already proxies established for this session), its joining policy, its coupling mode, the resources on which work is being conducted, the tools being used, etc. A session record can be used by client to find the sessions matching certain criteria (e.g., the session that are publicly accessible, and in which Fred is editing the file "paper.doc"). A session state contains other information only of interest to members of the session (e.g., an action list for undo operations).

Each member of a session has a copy of the session state, as does the session service. Session state fields may be either shared or private. Shared fields have identical values for all copies (e.g., the coupling mode of the session) in the same session, while private fields have separate copies for each state object (e.g., the undo list). Orthogonally, some fields may be marked as persistent, while others may be transient. Transient fields are discarded before saving state objects back to the persistent store service.

Normally, applications first query the session service to find an appropriate session. Then, they contact a server and create a new session or join an existing one using the connection protocol (see below). The contacted server may need to set up a new session proxy object and create a new state object with the session service. It then makes a new instance of the session state object and provides this to the new virtual client that corresponds to the application. The application may update its copy of the state, which may lead to changes in the session state copy in the session service (which may then lead to notifications to other members of the session).

3.2.2.4 Connection service

As stated above, the connection service is used by clients to create, delete, join, or leave sessions. Note that this very lightweight service runs between the client and some server *that is not the session server*. When a server receives a connection service request for the creation of a new session, it uses the session service described above to create a new session. It then responds to the requesting client with the appropriate information.

3.3 Applications

As mentioned above, there are three basic kinds of applications in a CB-OHS. Firstly, applications may be built from scratch or modified to participate in the CB-OHS. This is the least common way to integrate applications, since it is generally the most costly with respect to programming effort. An example of such an application is the Session Viewer, which allows users to browse a graphical representation of the session records available in a given session server.

Secondly, applications may be "wrapped" by other processes or code to communicate with servers. This is the most common approach for integrating third party applications into a CB-OHS. Literature on the details can be found in other sources (e.g., [Whitehead 1997]). In such cases, the wrapper may take the form of a full fledged process that may even be distributed from the server and wrapped client to a set of macros or other extensions executed by the application process. Some examples of such applications we have wrapped include Emacs, MS Word, and the Netscape Web browser.

Thirdly, applications may be run inside an integration framework which has itself been integrated into the CB-OHS by either of the above two options. This option bears some resemblance to the wrapper approach to integration, except that the "wrapper" (framework) in this case brokers the interaction between several clients and (potentially) several servers. This bears much resemblance to the "client side function" approaches discussed in the OHSWG literature [Anderson et al. 1997, Nürnberg and Leggett 1997].

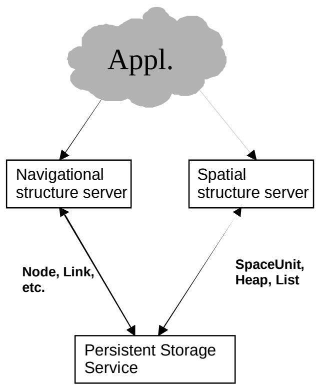
4 Integration options

This section describes a variety of integration options provided by the CB-OHS framework being introduced in this paper. The need to integrate two or more structure services typically occurs when a hypermedia designer wishes to provide several structuring mechanisms for the same body of hypermedia *content*. This content may be a large set of engineering documentation, botanical data or public Web pages on a specific subject. The essential issue here is thus to view and manipulate the different types of structures over the *same* body of content. By integration of structure services, we mean to achieve a smaller set of structure abstractions than we have when we operate with the union of abstractions provided by two independent structure services. The CB-OHS infrastructure allows a variety of integration policies. We see several integration options for each of the layers in our framework. Furthermore these can be combined to each others. We do not claim these options to be comprehensive, but they illustrate the span of possible integrations options. The integration possibilities are valid for arbitrary structure services and applications, but in this section, we choose to illustrate the possibilities by discussing the integration between navigational (see Section 3.2.2.1) and spatial (see Section 3.2.2.2) structure services.

4.1 Backend layer integration

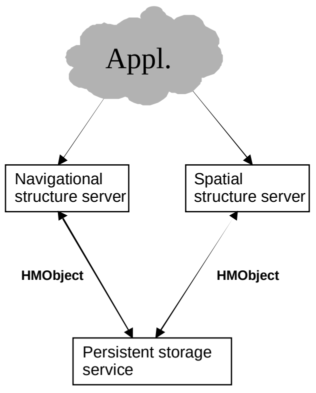
The basic persistent storage service stores hypermedia objects persistently and provides a number of collaboration and versioning services to arbitrary hypermedia objects. The basic SBMS handles generic storage units, as described in Section 3.2.1.1, which are generalized hypermedia objects that can be assigned various characteristics. When a structure server wishes to store a special hypermedia object, it is translated into the generic hypermedia object with enough characteristics to enable the structure server to reestablish the specialized object when retrieved. The basic SBMS may be utilized to provide the two integration options, discussed below.

4.1.1 Multi-typed storage abstractions



The first backend integration approach is to let each structure service represent a specific unit of content with its favorite content structuring concept. In the navigational structure server content units are typically represented with *nodes*, and in a spatial structure server content is typically represented by means of *SpaceUnits* [Bucka-Lassen et al. 1998]. The simplest possible integration is to let each structure service create its favorite abstraction for a specific Web page, for example, and store two different objects representing essentially the same content. The user will experience the different structuring mechanisms as separate layers on top of the same content, and the structures per se may never integrate. This approach, however, implies redundant representations, in which two or more different structure abstractions represent the same content. Moreover, it is difficult to perform combined computations such as queries or graphical visualizations on the union of the structures.

4.1.2 Storage abstractions with multiple views

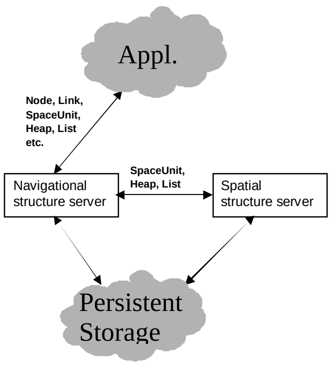


The second backend integration approach removes the redundancy in the first approach. Structure servers are allowed to "annotate" the structures created by other structure servers with additional type and characteristics information allowing the structure servers to maintain their own individual views on shared objects. In this way, a navigational server may store a node representing a certain piece of content specified with a ContentSpec. When a user wishes to structure this piece of content with at spatial hypermedia tool, the storage unit for the node is identified from the ContentSpec, and the corresponding storage unit is now annotated with type and characteristics information that allow the spatial structure server to view and manipulate it as a SpaceUnit simultaneously with a navigational structure server viewing it as a node. This approach reduces redundancy at the backend, but it complicates the creation of objects, since the persistent storage service needs to be checked for other structures representing the same ContentSpec, for example.

4.2 Middleware layer integration

The middleware layer consists of structure services that can manipulate their specific structuring abstractions. A structure service typically provides the behavior and policies associated with a certain type of structure. It thus mediates between the generic storage representation and the specific format and behavior that is needed by an application. When it is necessary to integrate two or more structuring abstractions, the middleware layer may provide two different kinds of integrations for the application layer, discussed below.

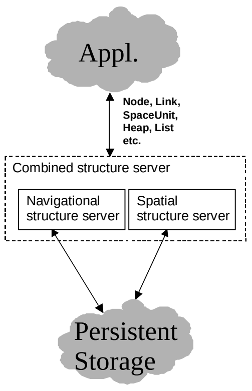
4.2.1 Inter-server communication



The first approach to providing middleware integration of services is to let the two servers communicate in order to provide a combined service to the application layer. In this approach, one of the structure servers is appointed as the master, and gets the responsibility for tunneling the structures provided by other structure servers to the applications at the frontend layer. This implies that one of the structure servers, say the Navigational server, needs to be extended with a module that communicates with other structure servers. The core may have to be modified as well to make a semantic integration between the different structures.

4.2.2 Multiple service servers

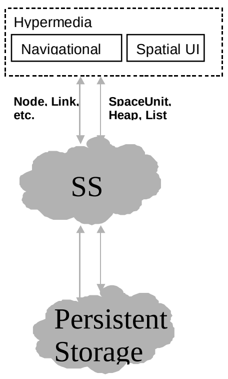
The second approach is based on the idea of wrapping multiple structure servers into a combined structure server that responds at least to the union of the interfaces provided by the individual structure servers. However, more semantic integration can be built into the combined server if needed. The advantage of this approach is that the individual structure servers remain unmodified and they can be used in other contexts as well.


Appl.
Node, Link, SpaceUnit, Heap, List etc.
Combined structure server
Navigational structure server
Spatial structure server
Persistent Storage

4.3 Frontend layer integration

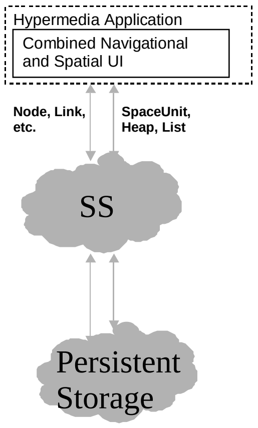
The frontend layer is the least standardized in the CB-OHS infrastructure. The frontend layer is highly dependent on the nature of the application domain in question (i.e., the types of tools to integrate with as well as requirements for the look and feel of the UI). The central issue of integrating third party applications is discussed in Section 3.3. Such integrations are well explored in the OHS literature. However, the notion of integrating different structure services (i.e., different structure abstractions) and integrating third party applications at the same time has not been explored much until now. We see two different integration options at this layer as well.

4.3.1 Application integration



The first approach is to provide separate user interfaces for each of the structure services and have them integrated in the same CSF or wrapper. This approach has been advocated and implemented in the Arakne framework [Bouvin 1999]. The advantage is that common functionality can be provided by the environment, while the special structure functionality is provided by the separate UI component. For instance, we can have a window with the node and link maps and another window with the spatial organizer tool. The advantage of this approach is that the hypermedia client can be written in a component architecture (e.g., Java beans), in which new structuring mechanisms are plugged in seamlessly into the framework.

4.3.2 Interface integration



The second approach integrates all of the different structure service functions in one unified user interface. This requires a uniform representation of the integrated (smaller set) of structure abstractions and instant access (via menus and toolbars) to the relevant structuring operations from the integrated set of available structure services. For instance, a spatial editor window may be the main interface to the hypermedia structures, and various linking and annotation operations may be invoked directly on the space units displayed in the organizer window. The advantage of this approach is a more homogeneous interface. However, extensibility with new structuring mechanisms is harder, since additions are harder to make modular than in the first approach.

4.4 An optimal integration approach?

As pointed out previously, the above options each have their advantages and disadvantages, but it is hard in general to recommend a specific combination of integration options. Determining the optimal choice of integrations may depend on both the application domain and the set of structure services in question. In the experiments we have undertaken so far, we have chosen the following approach for integrating the navigational and spatial structure services. At the frontend layer, we are using the *Application integration* approach, in which each structure service has its own recognizable user interface. At the middleware layer, we are using the *Multiple service servers* approach, in which the different structure services are integrated via common server that has a "protocol ring" that dispatches the operations to the relevant structure services. At the backend layer, we are using the *Multi-typed storage abstractions* approach, in which the storage units have type attributes that enable both the navigational and the spatial structure service to recognize the objects an interpret them with the semantics specific to the particular structure service.

The implications of these choices are that we have no storage redundancy, we have the structure services integrated in on single server, but we preserve the user interface functionality in a separate view for each of the structure services. We find this approach optimal at the Backend and Middleware layers, but at the Frontend layer, a specific application domain may require a more tight integration of the functionality. This can, however, be provided as an interface integration on top of the chosen approach for Middleware layer and Backend layer integration.

5 Related work

There is in particular two areas of research that relates to work reported in this paper: document authoring and structuring; and, integration of hypermedia structures. This section briefly discusses how the approach taken in this paper advances the state of the art in these areas.

5.1 Document authoring

A number of hypermedia systems such as NoteCards [Halasz et al. 1987], SEPIA [Streitz et al. 1991], Aquanet [Marshall et al. 1991], and VIKI [Marshall and Shipman 1995] provide spatial support for brainstorming, sorting and organizing iconic representations of information in a 2 or 2.5 dimensional space. The general hypermedia systems NoteCards and Sepia, for example, provide this kind of support in their graphical browsers for making placeholders for text to be written in corresponding nodes of the hypermedia structure. The pure spatial hypermedia systems Aquanet and VIKI provide support for sorting and organizing labeled icons on a 2.5 dimensional space, in which relations among the represented information units are inferred from the proximity between objects in the spaces and the icons visual characteristics. The hypermedia structures are generated by a spatial parser similar to the one we described above. The resulting structures are a kind of hypermedia composite as proposed by [Halasz 1988].

The CB-OHS approach discussed in this paper builds on these ideas for supporting the scenario in Section 2. However, with this approach, it is possible to integrate users' favorite editing tools to edit the content of the hypermedia nodes and provide the external linking mechanisms provided by open hypermedia services using a Web infrastructure. At the same time the spatial server outputs full-fledged composites that can be kept transient or be stored persistently and be target for structuring like any other hypermedia node.

5.2 Service integration

The basic link is not the only relationship among information units that has been applied in the hypermedia field. Many structures such as composites, guided tours, taxonomies, spatial structures, work flows, synchronization, Toulmin structures, etc., have been explored. Common to most of this work, however, is that linking mechanisms should be available and co-exist with the other structuring mechanisms being used. Thus, the efficient and effective integration of different structuring mechanisms is becoming an important hypermedia research issue [Nürnberg et al. 1998]. We see at least two different approaches for providing integrated structures: retrofitting or translating the new structures to well-known hypermedia structures such as links and nodes; and, making a hypermedia services for each type of structure and then combining those services.

5.2.1 Retrofitting or translating

An example of this category is the work by Wang and Haake [1998], in which they discuss workflow support and describe the CHIPS system, in which workflow objects are retrofitted to exist as links and nodes which have been augmented with special features. Another example is the work by Shipman et al. [1998], in which a spatial system (VIKI [Marshall and Shipman 1995]) is used as a front end for editing guided tours (Walden's Paths [Shipman et al. 1998]). In this case, the spatial structures are translated by means of an interchange format to become guided tour structures. We claim that this approach in which all possible new structures need to be mapped to traditional hypermedia structures limits the extent to which advanced hypermedia support can develop in the future. The CB-OHS approach is meant to avoid retrofitting and translation.

5.2.2 Combining structures

An example of this category is the work by Nürnberg et al [1996b], in which hypermedia support for botanist is provided with a tool called TaxEd, which is built on top of the HOSS [Nürnberg et al. 1996a] system. The HOSS approach is to atomize hypermedia structures into composable primitives, which in turn are supported by structure servers in the HOSS architecture. Linking is supported in a NavMan structure server, annotations in an AnnoMan structure server, taxonomies in a TaxMan structure server and so forth. The consequence of this HOSS approach is that an end user application such as TaxEd becomes quite complicated because they have to handle three orthogonal set of structures managed by their own structure servers that do not share any information. The advantage of this approach over cases described in the previous section is that it is open for an arbitrary rich set of new structures.

Another example in this category is the Devise Hypermedia (DHM) framework [Grønbæk and Trigg 1994; Grønbæk 1994], which provides an abstract and general object oriented framework in which new hypermedia structures can be implemented as specializations of general classes. In this framework, links, annotations, composites, guided tours, etc. co-exist in the same framework and can thus be combined in arbitrary ways be provided to applications in a uniform manner. Compared to the HOSS approach, the object-oriented DHM framework approach results in a single structure server combining all types of structures, which is less flexible with respect to extension.

The approach discussed in this paper builds on the philosophy of combining structures. We have combined the HOSS and DHM approaches by providing multiple structure servers which share a common data model as proposed by Grønbæk [1998]. In this way, we achieve flexibility and extensibility as well as the ability for structure servers to share and reference each others structures. This, in turn, makes it easier to write end user applications that combine structures and services.

6 Future Work

There are a number of steps we are in the process of undertaking to further the work presented here. Firstly, we are planning a deployment of this software to support the newsletter production tasks as described in the scenario in Section 2. We expect useful feedback on our initial prototypes from this field deployment.

Secondly, we are planning to use our spatial hypermedia server experiences that we have gained to form the basis of a draft standard proposal to the OHSWG for defining a spatial hypermedia interface. We hope that our initial findings on the integration of navigational and spatial hypermedia within a common framework, combined with our experiences in drafting and implementing the current OHSWG navigational interface standard, will enable us to generate a powerful and compatible interface specification after the first round of field testing and subsequent modification.

Thirdly, we would like to examine the possibilities of session semantics a cross different types of structure servers. Currently, the session semantics defined by the OHSWG and those implemented in our prototypes are all within the scope of a single type of structure server (even if across multiple instances of this type). However, since we are now considering collaborative work that spans structure server types, we feel it is necessary that some functionality be implemented that allows different structure server types to cooperate within a session. We expect our experiences with our field deployment will provide a good starting point for defining what such types of semantics might be.

Fourthly, a number of more minor improvements to the infrastructure are being considered or are already underway. We are looking at increasing the use of off-the-shelf component-based frameworks for offloading some of the communication work currently performed by the infrastructure itself.

Although we have performed some very basic tests with integrating RMI into our SBMS, we are still using TCP-based communication between all our entities. We are also considering porting our location and naming services to the Uniform Resource Name draft standard [IETF 1997].

7 Conclusions

Since hypermedia concepts were introduced by Bush to address associative storage and recall tasks [1945], hypermedia has been applied to an ever widening array of other problems, such as information analysis, classification, argumentation support. Traditionally, these new problems have called for a new set of structural abstractions, resulting in new conceptual structure services often realized in monolithic or closed systems. However, many problems, such as the collaborative editing problem described above, require structural abstractions from several of the aforementioned domains. In the collaborative editing example, we showed the desirability of both spatial hypertext (information analysis) and navigational hypertext (associative storage and recall) abstractions. Traditionally, it has been difficult to integrate abstractions from several different domains in a way that allows meaningful interoperation.

Here, we proposed a component-based open hypermedia system (CB-OHS) solution to the problem of integrating several different structure services. We showed many different options for effecting such integrations, and described the relative advantages and drawbacks of each. We also compared these CB-OHS solutions to those approaches used by other, non-CB-OHS systems, such as retrofitting or translating, and described the potential drawbacks of these latter two approaches. We feel that CB-OHS's have shown themselves to be a promising platform for more work into inter-domain interoperability.

References

Anderson, K., Taylor, R., and Whitehead, E. 1997. A critique of the open hypermedia protocol. *Journal of Digital Information 1* (2).

Berners-Lee, T., Cailliau, R., Luotonen, A., Nielsen, H. F., and Secret, A. 1994. The World Wide Web. *Communications of the ACM 37*(8), 76-82.

Bernstein, M., Bolter, J., Joyce, M., and Mylonas, E. 1991. Architectures for volatile hypertext. *Proceedings of the Third ACM Conference on Hypertext (HT '91)* (San Antonio, TX, Dec), 243-260.

Bolter, J. and Joyce, M. 1987. Hypertext and creative writing. *Proceedings of the Second ACM Conference on Hypertext (HT '87)*, (Chapel Hill, NC, Nov), 41-50.

Bouvin, N. 1999. Unifying strategies for Web augmentation. *Proceedings of the Eleventh ACM Conference on Hypertext (Hypertext '99)* (Darmstadt, Germany, Feb).

Bucka-Lassen, D., Reinert, O., and Pedersen, C. A. 1998. CAOS — Cooperative Authoring using Open Spatial hypermedia. http://www.daimi.aau.dk/~oreinert/speciale/.

Bødker, S. 1991. *Through the Interface — a Human Activity Approach to User Interface Design.* Lawrence Erlbaum Associaties, Hillsdale, NJ.

Bouvin, N. O. 1998. Strategies for Web augmentation. submitted to Hypertext '99.

CIT: Danish National Center for IT Research. 1999. Coconut home page. http://www.cit.dk/coconut/.

Conklin, J. and Begeman, M. 1987. gIBIS: A hypertext tool for design deliberation. *Proceedings of the First ACM Conference on Hypertext (HT '87)* (Chapel Hill, NC, Nov). 247-251.

Davis, H. C., Knight, S., and Hall, W. 1994. Light hypermedia link services: A study of third-party application integration. *Proceedings of European Conference on Hypermedia Technologies.* ACM Press. 41-50.

Fischer, G., McCall, R., and Morch, A. 1989. JANUS: Integrating hypertext with a knowledge-based design environment. *Proceedings of the Second ACM Conference on Hypertext (HT '89)* (Pittsburgh, PA, Nov). 105-117.

Grønbæk, K. 1994. Composites in a Dexter-based hypermedia framework. *Proceedings of the ACM European Conference on Hypermedia Technology (ECHT '94)* (Edinburgh, UK, Sep). 59-69.

Grønbæk, K., Hem, J. A., Madsen, O. L., Sloth, L. 1994. Cooperative hypermedia systems: A Dexter-based architecture. *Communications of the ACM 37*(2) (Feb). 64-75.

Grønbæk, K. and Trigg, R. H. 1996. Toward a Dexter-based model for open hypermedia: Unifying embedded references and link objects. *Proceedings of the Seventh ACM Conference on Hypertext (HT '96)* (Washington, DC, Mar).

Grønbæk, K. 1998. OHS interoperability: Issues beyond the protocol. *Proceedings of the Fourth Workshop on Open Hypermedia Systems.* Technical report CS-98-01 Aalborg University Esbjerg.

Halasz, F. G., Moran, T. P., and Trigg, R. H. 1987. NoteCards in a nutshell. *Proceedings of the ACM CHI+GI '87 Conference* (Toronto, Canada, Apr). 45-52.

Halasz, F. G. 1988. Reflections on NoteCards: Seven issues for the next generation of hypermedia systems. *Communications of the ACM 31*(1). 836-852.

IETF. Uniform Resource Name syntax. Internet Engineering Task Force request for comments 2141(proposed standard) (May).

Jühne, J. Jensen, A. T., and Grønbæk, K. 1998. Ariadne: A Java-based guided tour system for the World Wide Web. *Proceedings of the WWW 7 Conference* (Brisbane, Australia).

Kaplan, N. and Moulthrop, S. 1994. Where no mind has gone before: ontological design for virtual spaces. *Proceedings of the ECHT 94 European Conference on Hypermedia Technology (ECHT '94)* (Edinburgh, Scotland, Sep) 206-216.

Landow, G. and Kahn, P. 1992. Where's the hypertext? The Dickens Web as system-independent hypertext. *Proceedings of the ECHT '92 ACM Conference on Hypertext (ECHT '92)* (Milano, Italy, Dec) 149-160.

Marshall, C., Halasz, F., Rogers, R., and Janssen, W. 1991. Aquanet: a hypertext tool to hold your knowledge in place. *Proceedings of the Third ACM Conference on Hypertext (HT '91)* (San Antonio, TX, Dec). 261-275

Marshall, C., Shipman, F., and Coombs, J. 1994. VIKI: spatial hypertext supporting emergent structure. *Proceedings of the ECHT '94 European Conference on Hypermedia Technologies* (Edinburgh, Scotland, Sep). 13-23.

Marshall, C. and Shipman, F. 1995. Spatial hypertext: designing for change. *Communications of the ACM 38* (8). 88-97.

Marshall C. and Shipman, F. 1997. Spatial hypertext and the practice of information. *Proceedings of the Tenth ACM Conference on Hypertext (Hypertext '97)* (Southampton, UK, Apr). 124-133.

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

Michalak, S. and Coney, M. 1993. Hypertext and the author/reader dialogue. *Proceedings of the Fifth ACM Conference on Hypertext (HT '93)* (Seattle, WA, Nov) 174-182.

Moulthrop, S. 1989. Hypertext and "the hyperreal". *Proceedings of the Second ACM Conference on Hypertext (HT '89)* (Pittsburgh, PA, Nov) 259-268.

Nürnberg, P. J., Schneider, E. R., and Leggett, J. L. 1996a. Designing digital libraries for the hyper-literate age. *Journal of Universal Computer Science 2* (9).

Nürnberg, P. J., Leggett, J. J., Schneider, E. R., and Schnase, J. L. 1996b. HOSS: a new paradigm for computing. *Proceedings of the Seventh ACM Conference on Hypertext (HT '96)* (Washington, DC, Mar), 194-202.

Nürnberg, P. J., Leggett, J. J., and Schneider, E. R. 1997. As we should have thought. *Proceedings of the Eighth ACM Conference on Hypertext (HT 97)* (Southampton, UK, Apr).

Nürnberg, P. J., and Leggett, J. J. 1997. A vision for open hypermedia systems. *Journal of Digital Information 1* (2).

Nürnberg, P. J., Leggett, J. J., and Wiil, U. K. 1998. An agenda for open hypermedia research. *Proceedings of the Ninth ACM Conference on Hypertext (HT 98)* (Pittsburgh, PA, Jun). 198-206.

Open Hypermedia Systems Working Group home page. 1997. <http://www.ohswg.org/>.

Parunak, H. 1993. Hypercubes grow on trees (and other observations from the land of hypersets). In *Proceedings of the Fifth ACM Conference on Hypertext (HT '93)* (Seattle, WA, Nov) pp. 73-81.

Schuler, W. and Smith, J. 1990. Author's Argumentation Assistant (AAA): A hypertext-based authoring tool for argumentative texts. *Proceedings of the First European Conference on Hypertext (ECHT '90)* (Versailles, France, Nov). 137-151.

Shipman, F. M., Furuta, R., Brenner, D., Chung, C., and Hsieh, H. 1998. Using paths in the classroom: Experiences and adaptations. *Proceedings of the Ninth ACM Conference on Hypertext (HT 98)* (Pittsburgh, PA, Jun). 267-276.

Smolensky, P., Fox, B., King, R., and Lewis, C. 1988. Computer-aided reasoned discourse or, how to argue with a computer. *Cognitive Science and its Application for Human-Computer Interaction*, R. Guindon, Ed. Ablex, Norwood, NJ. 109-162.

Streitz, N., Hannemann, J., Lemke, A., Schuler, W., Schütt, H., and Thüring, M. 1992. SEPIA: A collaborative hypermedia authoring environment. *Proceedings of the European Conference on Hypertext 1992 (ECHT '92)* (Milan, Italy, Nov-Dec). 11-22.

Trigg, R. H. 1988. Guided tours and tabletops: Tools for communicating in a hypertext environment. *Transactions on Office Information Systems 6*(4) (Oct). 398-414.

Wang, W. and Haake, J. 1998. Flexible coordination with cooperative hypermedia. *Proceedings of the Ninth ACM Conference on Hypertext (HT 98)* (Pittsburgh, PA, Jun). 245-255.

Whitehead, E. J. 1997. An architectural model for application integration in open hyermedia systems. *Proceedings of the Tenth ACM Conference on Hypertext (Hypertext '97)* (Southampton, UK, Apr). 1-12.

Wiil, U. K. (ed.) 1998. *Proceedings of the Fourth Workshop on Open Hypermedia Systems.* Technical report CS-98-01 Aalborg University Esbjerg.

Zellweger, P. T. 1989. Scripted documents: A hypertext path mechanism. *Proceedings of the Second ACM Conference on Hypertext (Hypertext '89)* (Pittsburgh, PA, Nov). 1-26.

Østerbye, K. and Wiil, U. K. 1996. The Flag taxonomy of open hypermedia systems. *Proceedings of Seventh ACM Conference on Hypertext (Hypertext '96)* (Washington, DC, May). 129-139.