

A Network Based Collaborative Authoring System

*Michael B. Spring, Bordin Sapsomboon,
Wasu Chaopanon, and Chajong Kim*

Department of Information Science and
Telecommunications
University of Pittsburgh
Pittsburgh PA 15260
(spring@sis.pitt.edu, www.sis.pitt.edu/~cascade)

ABSTRACT

This paper describes the design principles and structure of a prototype system for collaborative authoring over a wide area network. The operational prototype provides a test bed for the examination of human computer interaction, group interaction, group support, document structures, and network protocols. The paper begins with a description of the problem and the history of efforts to address it. The system includes extensive support for commenting and comment review, facilities for document space navigation, and tools for controlling and monitoring work group activity, including document locking and activity recording.

Keywords

Collaborative authoring, groupware, augmentation, visualization, structured document, networked application

INTRODUCTION

Structured documents, ubiquitous networks, client-server protocols, and advanced human-computer interfaces all offer the potential for allowing new kinds of systems to be built to support collaboration. In 1960, J.C.R. Licklider[24] suggested human-computer symbiosis as a premise for research. Engelbart[10] developed the Augmentation Research Center to explore systems to augment human intellectual activity. The World Wide Web has dramatically changed how people see electronic publishing and structured electronic documents. Berners-Lee [4] envisioned the prototype he developed at CERN as an authoring tool as well as a viewer. Despite this, the integrated use of logical copy-marking and distributed hypertext systems for document creation is not well developed. We still tend to create documents in isolation.

This paper describes a design process and prototype for collaborative authoring over a wide area network. As with

any system design, there are choices made. The more critical are mentioned here to provide context. First, the system uses a cross platform RPC like protocol for communication. Our goal in using this protocol is to allow us to rapidly define and modify the interface between the client and the server. Once a stable set of interfaces is developed they may be incorporated as a set of interfaces into a protocol such as CORBA. Thus while the system operates on the Internet, it does not use the HTTP protocol and thus strictly cannot be considered a "web" application. On the other hand, we see no reason why a stable version of the interfaces could not be tunneled through the HTTP-NG protocol using CORBA. Second, we have chosen a mixed model for storage that combines simple file storage, tag based meta information, and DBMS records. For example, by using a DBMS as an anchor repository, it is possible to view comments on a document as point functions and thereby allow comments to be made on a document simultaneously by multiple people. The choices about storage form are made on an opportunistic basis with an eye to having the ability to produce archival forms for dissemination. Finally, choices about the design of the client interface have been made with an eye to providing an augmented interface that relies heavily on visualization and visual cues. It is our intent to explore how collaborative authoring in an electronic environment might be made "simple" for users. Thus the ultimate goal of the effort is to explore how a system for collaborative authoring can be made attractive to users. We believe that we have uncovered a reasonable interface definition for the client and server components and have developed some insights into how to structure the document store using concepts from hypertext, structured documents, and database approaches.

Motivation and Scope of Application

A system for collaborative authoring presents problems not encountered in the development of a system for viewing documents across the Web. It is not surprising that early browsers delimited the scope of the problem they addressed to viewing. Collaborative authoring comes in many flavors (small group, large group, single document, multiple documents, etc). This paper focuses on large group efforts such as might be typified by the following scenarios:

1. the development of state or federal legislation,
2. the development and balloting of standards, and
3. the development of strategic plans.

In these situations, the goal is the production of a document. We would distinguish this from a collaborative workflow system in which documents are processed to achieve some other goal -- approval of a loan, or production of a product. These are all situations in which

Space for ACM copyright information.
Remember to delete this before
submitting final version. (Use a
column break in MS word to stop text
from overwriting this area.)

some of the work may be done synchronously and some may be done asynchronously. There are both face to face and electronic meetings. The system needs to operate across a WAN.

It has occurred to us that a system such as the one we have developed may also be applicable to situations such as the peer review of proposals submitted for funding or articles submitted to journals. At the outer limits, systems based on the principles we have used may be applicable in situations where the number of authors is rather small -- e.g. the development of articles. The primary application remains large projects where long periods of time and geographic dispersion make coordination difficult and where small group type interpersonal communication is likely to fail.

The optimal scenario is a multi-year authoring and review process with tens to hundreds of people such as might be the case for the development of a national or international information technology standard. The cost of an information technology standard is about \$10,000,000 when all costs are considered and takes approximately seven years[42]. The system described in this paper might reduce this time and cost dramatically.

Background

A number of things can be learned from the literature. First, a number of people have written about the requirements for effective support of collaborative authoring, both theoretically and in terms of prototype systems. Second, there is a large literature which explores the potential of structured documents -- both tree structured (i.e., SGML) and mesh (i.e. hypertext). Finally, there is a growing literature on the design of enhanced interfaces and agents. All three of these literatures combine to support the design. A brief sample of some of the findings is presented below.

A system should support, not control, collaboration. "Computer systems to support the writing process will be of most help if they fit the writer's perception of the task and assist whichever strategy the writer chooses to adopt"[36]. Software must support a range of writing strategies and provide support to people, environment, and task [25]. Group roles, e.g. author, reader, etc. and interactions e.g. communication, collaboration, coordination, etc. must be supported. Sharples et al. [35] indicates the diversity of activities in collaborative authoring and the requirements for new generation collaborative writing tools.

Rimmershaw [33] indicates that comments and revisions need to be associated with the appropriate portion of text. He also indicates the need to facilitate making meta-comments and linking these comments to specific parts of text. Rimmershaw also addresses the need for a system that allows the text to be modified which implies the ability to identify revisors. Eklundh [7] focuses on the need for appropriate document views that enable the writer to perform tasks with limited cognitive overload.

Ellis, Gibbs, and Rein [8] define three broad support functions: communication, collaboration, coordination. Ede and Lunsford [6] note the importance of coordination. Neuwirth et al. [29] identify several issues related to asynchronous collaborative authoring. Commenting is a major activity for communication within authoring groups and between readers and such groups. Collaboration supports include social interaction support among co-authors and commenters and cognitive support for co-authoring and external commenting. Fish[15] explores the strategy of defining social roles, such as primary author, commenter, reader, for individual collaborators to reduce coordination problems.

A collaborative authoring system can be enhanced by hypertext. Hypertext structuring and link management can be based on theoretical constructs provided by the Dexter[17] and Amsterdam[18] hypertext models as well as the HyTime standard. Comment review systems can be implemented as hypertext. Navigation through a network of documents is facilitated by hypertext inspired browsers and visualization tools. A number of systems[1, 2, 16, 21, 26, 28, 34] have been built providing various services. Comments and revisions can be linked to text providing context. Information available from SGML structuring conventions and hypertext links can be used to aid collaboration and management.

At the interface level, a variety of navigational tools, can be used to overcome the difficulty of navigating large document spaces. There is great deal of work that has been done on browsers. Our work builds most on various texture maps and structure maps[19, 31, 37]. Spring and Jennings[39] articulate a series of rules for mapping abstract data to virtual spaces. Navigation tasks involve finding groups of objects of interest, finding specific objects of interest, following interesting paths, and exploration for objects of given attributes. Visualization uses preattentive stimuli to process large amounts of data. The current research uses a variety of visual navigation aids and tools in a document space that is semi-structured and populated with objects that have multiple attributes and multiple relationships. Various tools focus on the different navigation tasks[41].

Beyond these forms of augmentation, agents can be used to extend the capability of group members by offloading nonessential cognitive activities. Bentley et al.[3] have looked at how various agents, tools, and methods might be incorporated in a work environment to assist a group. Ferguson[14] suggests that agents work at three levels--reactive, planning, and modeling. We believe that 5 levels may be a more appropriate classing for collaborative environments. Executive agents are responsible for contributing planning level skills involving the entire group. Collaborative agents are responsible for independent action and direct contribution where the action involves more than one individual. Contributory agents are responsible for direct contribution to action where the action involves only a single individual.

A Network Based Collaborative Authoring System

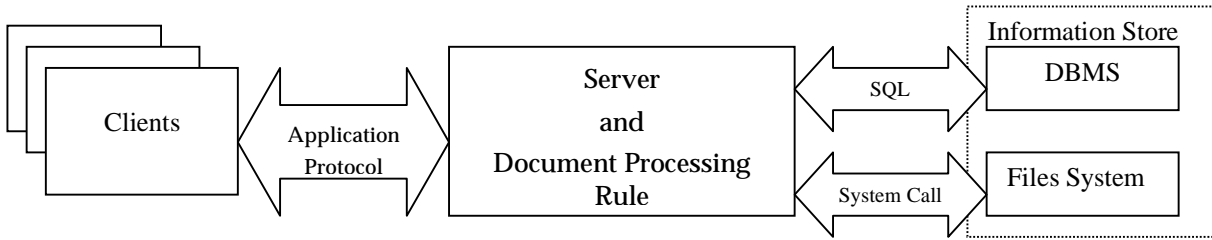


Figure 1: System Components

Communications agents are responsible for managing communications. Service agents are responsible for providing low level services to individuals or system components. Finally, a collaborative authoring system must provide cognitive support for peripheral authoring tasks. One important component of the peripheral tasks is support for social interaction[5, 40].

DESIGN

The design of CASCADE has involved two separable processes. The first involves decisions about how the information store will be modeled, controlled by the server, and communicated to the Client. The second process involves making this model visible (or invisible) to the user via the client interface. These two design efforts are discussed below.

System Design Principles

As shown in Figure 1, a net-based collaborative authoring system requires an information store, a server to manage the information store and a client that allows individuals to access the store. There must also be a set of interfaces in a protocol to link the client and server and a set of data tables to provide management functions, which we assume here to exist in a DBMS.

The goal of the design effort is to insure the applicability of the model to the problem at hand as well as the extensibility of the solution to new opportunities that may arise. Because the system is a research prototype, we have allowed for a more evolutionary design than would be the case for a production system. In addition, where possible, we have over built the system to provide hooks for research data gathering modules and to provide the opportunity to extend the system so as to allow others to mould it to some other problem.

The design goals for a collaborative authoring system include:

1. Provide a client interface that allows multiple users to access and manipulate the information store so as to:
 - create and edit documents,
 - manage individuals and groups,
 - navigate and find documents and components, and
 - gather information about project and user activity.
2. Provide an information store that will:
 - allow multiple classes, types, and subtypes of

- documents,
 - allow decomposition into subcomponents,
 - allow composition of parts into larger aggregates,
 - allow a variety of relationships between parts
 - allow transformed into different forms both internally and for import and export
 - be described in terms of various meta information.
3. Provide a server with the capability, information, and intelligence needed to control the information store and the users including the ability to:
 - model the information store at various levels,
 - facilitate the communication between group members, and
 - provide for controlled access to documents and document support functions,
 4. Provide an application protocol that will:
 - support communication simply and completely,
 - allow objects of any type to be moved,
 - maintain minimal state information,
 - be extensible, and
 - provide look-ahead and caching schemes that minimize network traffic.
 5. Provide a database schema to:
 - manage the document components,
 - store information about documents and components,
 - store information about users,
 - manage the component relationships, and
 - manage the user access to the information store.

Each of these goals carries a wealth of nuance. For example, the decomposition and aggregation of components allows respectively for locking at various levels of granularity and for the composition of structured documents from pieces. Each of these principles is built upon careful consideration of how the system might embrace and benefit from technological capability to allow collaboration functionality.

Interaction Design Principles

The design of an interface can involve three broad classes of design efforts: replication, reengineering, and innovation. Replication involves the design of a system where existing systems have pretty much defined a standard for human computer interaction -- e.g.

spreadsheets and other metaphors. In this case, designs should closely approximate existing practice. When a process is dramatically redefined and the division of labor between human and machines takes on dramatically new proportions, we are talking about reengineering. This involves the redefinition of the tasks. At the same time, the particular interactions will probably still take on traditional forms. Occasionally, the interaction requires the design of interactions of radically new form. For example, one might suggest that "electronic hallways" [5] have yet to find any definitive form. When a completely new task is defined for the first time, innovative design is required. Innovation moves beyond reengineering and asks how new goals may be achieved via a symbiotic relationship between the humans and the system where, as Engelbart put it, the skill repertoires of human and system are redefined[11, 13]. Collaborative authoring, visualization of structured hypertext documents, navigation, and activity analysis have innovative design. In the design of these new capabilities the authors relied heavily on visualization, ad hoc hypertext structures using captured information streams, and autonomous communicating agents.

The client should be manifest as a part of a desktop environment within which the user works. This means that at any given time, a number of documents, containers, and tools may be open on the "desktop" with data moving freely between them. The tools that are found on the desktop will be both active (working on behalf of the user with minimal user interaction and direction) and passive (working only as directed by the user). As suggested by Kay[22, pp 199-200.], the metaphor should not constrain action, but suggest and encourage it. When the metaphor grows too constraining, magic or illusion should be the controlling principle. The system should provide more capability than the metaphor. Operationally, four concepts guided the interaction design: augmentation, visualization, information, and substitution.

Augmentation involves shifting "housekeeping" tasks to the computer. Any process consists of a set of tasks of which only a small set actually require human action and decision making. Various analyses of document processes can be used to identify subtasks requiring cognitive processing. For example, in some systems, an electronic comment requires 10-15 actions (depending on how the process is defined). Only one of these tasks, writing the comment itself, actually requires the commenter--the rest can be done by the system. "Augmentation"[9, 10, 12, 13] involves task analysis and redesign to create a set of processes that optimize the human side of the human computer interaction. We call this kind of redesign "augmentation" in the sense that the term was first used by Engelbart .

Visualization can increase the speed with which targets of opportunity can be addressed [23, 30, 32, 37, 38]. Where appropriate, the system should make use of visualization tools for information finding, navigation, and other tasks where visualization of data attributes can provide quicker

processing. Some of the tools developed for the system allow novice user to find a small set of documents out of 2000-3000 in only a second or two[20, 27, 41].

Information or informing, according to Zuboff[43] is a process where an data stream generated by a computer is used to improve the process. In collaborative authoring, as an example, mailing a revision to a group of co-authors can take up a great deal of time. It shouldn't however, because the system "knows" who is involved, how they can be contacted, what document was just worked on, and what review means. Thus, a mail function can begin with the "Subject","From", and "To" fields filled in. Clicking one button should attach the document. The user simply types the note and the document is off. This kind of interaction is based on "information"[43].

Substitution is the term we use for integrated agents. There are many things that we would like others to do for us. For example, a communication agent might keep track of where people "live" when they are using a system. Knowing that, the agent can go check those places when someone wants to contact them. When found, it let's the user know, and if the timing is still right, set up an interactive session. This kind of interaction is "substitution" in the sense that the agent undertakes some task on behalf of the user and substitutes its effort.

A PROTOTYPE SYSTEM: CASCADE

A system, called CASCADE, has been built using these principles. In addition to its operational capabilities, hooks have been built in the system to support research on augmenting authoring, editorial, and other document processes using the information available from structured electronic documents. Operationally, CASCADE reduces cognitive overhead in authoring of structured documents by (1) presenting information visually, (2) employing information streams to inform the process, (3) augmenting processes to as great an extent as possible, and (4) using software agents.

It reduces the complexity of document management by using a DBMS as a repository for shared group information and document hypertext information based upon structured document data. The DBMS is also used to assure document integrity and availability.

A Brief History of the CASCADE effort

Development of CASCADE began in 1993 as exercise in interface design. The first implementation was written for X Windows and provided common browsing interface to files and scripts providing information about lab schedules, software packages, course offerings, etc. In 1994, we began to explore how it might be developed as an augmentation research test bed. The project was renamed "CASCADE" – Computer Augmented Support for Collaborative Authoring and Document Editing. The system grew to include a large number of features driven by student interest, research papers, and problem definition. The system used the Unix file system and flat files for meta information

A Network Based Collaborative Authoring System

In 1997, the National Institute of Standards and Technology (NIST) provided support to study alternatives for collaborative authoring and to begin development a client server version of the prototype. In 1998, additional funding was provided to complete the client server version and begin field tests. It is this current version which is described below.

ARCHITECTURE OF CASCADE

CASCADE is a client server system. Clients run on multiple platforms making as much use as possible of plug-ins and extensions. A rich RPC based application protocol provides application partitioning and close to real time performance. The server controls a structured document information store supplemented by DBMS data stores. The client, the protocol, and the server-information store are described below.

It is sometimes difficult to picture how a system is implemented without seeing and using it. The CASCADE web site provides access to a tour of the system including dozens of the screens. We have included two composite screens here to give the reader some sense of the system. Figure 2 shows the main screen including the main menu bar and the document-folder navigation bar. The document window shows the comment links and the mural shows the location of all comments in the overall document. Comment and comment review windows are also open. The comment display window changes the color coding of the comments. Figure 3 shows two on the ad hoc hypertext constructions that may be called on to further aid navigation. The tree browser window shows the parent child relationships over any number of levels and may be used to navigate and display documents. The project activity menu shows the results of a query of document activity. It may be filtered and sorted with ease along any number of dimensions.

The Client

Each user sees the documents as his/her own. Selected tasks have been augmented so as to be as easy or easier than their counterparts in the physical world. The demands of coordinating collaboration are borne by the system. Efforts are made to maintain a standard look and feel in terms of types of dialogs; menu availability, placement and naming; scroll bar appearance and controls; activity semantics and terminology, etc. When features are new, every effort was made to bring them to the attention of the user as new. Below we briefly address three broad areas of functionality: navigation through document (work) space, document manipulation, and integration with other tools and systems.

Navigation

When the CASCADE client is started, the user chooses a server. CASCADE currently assumes that a project is contained on a server. A server houses any number of projects. After the user logs in to a server, a project selection dialog is presented allowing the user to select a given project. (This same dialog may be used during a

session to switch from project to project.) Once a user is given access to a project on a server, they may be given access to another project under the same password. (Of course there may be projects to which a given user does not have access.) A user may access only one project at a time. (This decision was made so that various variables

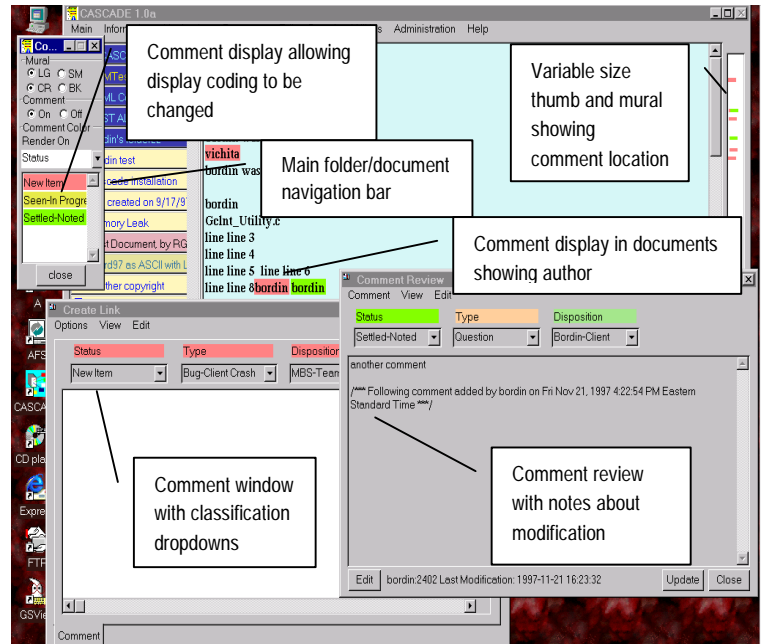


Figure 2: Main Screen with Comment Dialogs (Create, Review, Display)

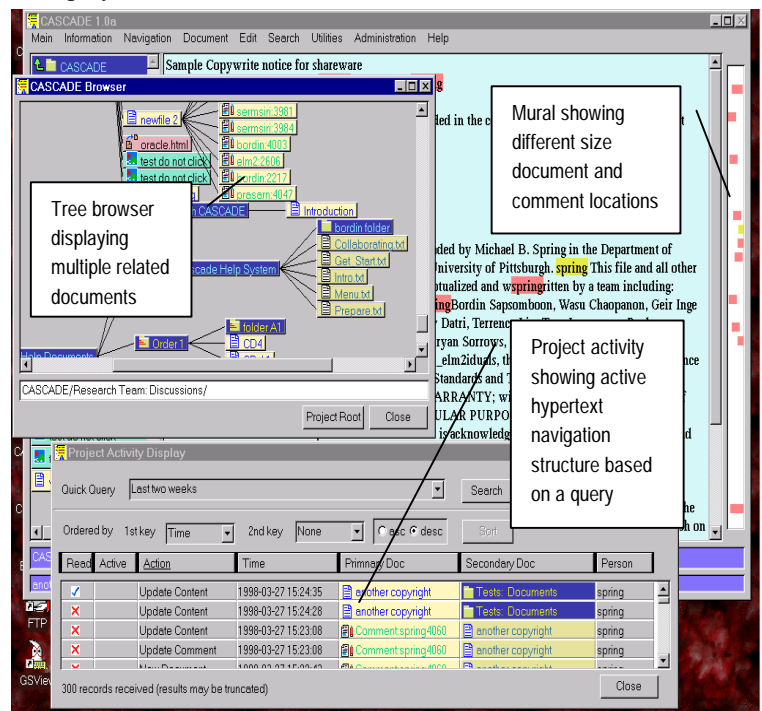


Figure 3: Main Screen with Tree Browser and Project Activity Hypertext Structures

by color.

CASCADE also provides query tools that enable the user to generate ad hoc hypertext structures showing documents and key related documents. For example, it is possible for the user to generate and store queries such as "all document edit actions in the last three months on documents x.y. and z" or "everything done by member of group y". This tool can be used for awareness "show me what has transpired recently" or to drill down on activities of a particular type, related to a particular document, or performed by a given individual or group. Reporting tools that summarize activity on a document or set of documents have also been prototyped.

Manipulation

Commenting and editing tools are at the core of CASCADE. Normally, users view documents in display mode, reading documents and making comments. It is assumed that for most people, the most time is spent in this mode. Comments are hypertext anchors that can respond to a single mouse click. The user must select edit mode to change documents, necessitated by the need to lock documents for editing. Commenting is an augmented activity in CASCADE. A single click names the comment node, links it to the source document, and informs the work group of its presence. The user simply needs to type the comment itself, change the default classifications if appropriate. After a comment is created, it is available for viewing by others.

This augmented commenting capability represents one of the dilemmas in CASCADE. Users would prefer to use their own editors to edit a document. However, to the extent that CASCADE is significantly better at adding and managing comment links than Word or WordPerfect, there is pressure to provide built-in viewers and editors. At this time, only basic ASCII editors are built in supporting the editing and linking capability. It is our intent to provide built in facilities for viewing and editing SGML documents with anchors/links as well. For all documents users may specify a plug-in editor/viewer.

Documents and folders can be created at any location where the user has permission to do so. There are many kinds of documents and folder that can be created and managed by CASCADE. Thus, order information may be required when a document is placed in an ordered folder. Within CASCADE documents and folders inherit any information that it makes sense to inherit from the parent. One of the more important things that a document inherits is its default protection and group access scheme. More on this feature of CASCADE can be found in the section on the server design.

Desktop Integration

Users have requested the ability to customize CASCADE to meet their personal preferences. CASCADE has been structured so that users may specify personal preferences for everything from colors and fonts to editors and display

software. CASCADE allows the user to specify a large number of personal preferences in a platform independent fashion. Everything that can be personally preferred without jeopardizing group semantics is under personal control. This includes access to and use of all software on the users platform from within the CASCADE context.

CASCADE provides a variety of support tools. For example, "User Activity" provide members of the project with a visualization of personal work schedules -- in anticipation of helping to set up times for synchronous meetings. "Comment analysis" provides a broad overview of where things are happening in the document space.

CASCADE also provides interfaces to desktop functions. For example, if a user wants to send a note to various members of the work group related to a document he or she is working on, CASCADE will open a dialog that will make some intelligent guesses about the recipients, originator, and subject of the mail note. While the user is free to edit these fields, the system makes every effort to make the task of mailing the note as simple as typing the message and saying "go".

Administration tools that set up projects, groups, passwords, and other administrative needs may also be accessed from the client rather than having to access specialized DBMS utilities. Desktop integration is minimal at the current time. CASCADE will be increasingly linked to utilities residing on both client and server via CORBA and IIOP interfaces.

The Communications Protocol

On the multi-platform version of CASCADE a proprietary RPC like protocol known as DAS -- Distributed Application Services -- is used. The protocol defines the client-server (application) interfaces. At the current time, about 45 data structures have been defined in the following areas:

1. Login/logout and setup messages allow exchanges between the client and the server that provide validation of a given user on a given server. The messages also provide information about current state of users, projects on a server for which that person is authorized and finally project environment information.
2. Node related messages includes creation and deletion of documents and folders (collectively referred to as nodes), editing and locking of nodes, meta information about nodes, and relationships -- parent and child as well as sibling, are all a part of this category. Nodes fall in two broad classes -- documents and folders. Folders may be ordered or unordered, and documents may be of a number of different types -- graphics, text, image, etc.
3. Comment related messages transfer specialized information about comments
4. Anchor related The anchor messages extend the capability to support collaborative authoring by

providing structures and tools for describing the hypertext links in a document without including them in the document per se.

5. Activity related messages allow for the transfer of information back and forth between the client and the server related to actions taken by the user.
6. Group and project related messages allow information about users, projects, groups and their relationships to be exchanged.

We continue to expand and redefine this list of messages as we gain experience with the exchanges. While our goal is to provide a simple, extensible, and robust set of idempotent messages, we continue to find situations in which speed demands more complex messaging aggregates and functionality requires judicious amounts of state information. This approach allows us to experimentally validate the messaging interface we are defining under actual usage. Ultimately, CASCADE will use a set of interfaces defined under CORBA such as might be provided by HTTP-NG.

The Server and Information Store

The server stores and controls access to the shared documents -- the information store. The server determines how the information store is presented. The server maintains information about documents and folders such as identification, node classification, node ordering information, ownership and protection in a DBMS. Other meta-information related to documents, users and authoring activities are also stored in the DBMS to ease the management of information for project, structured document, and hypertext modeling.

Five questions had to be answered in designing the server.

1. How will the information store be modeled?
2. What information will be maintained by the server?
3. How will server information be stored (DBMS or file system)?
4. How will access to the information be controlled?
5. What processing will the server be responsible for?

How will the information store be modeled?

The more important server function is modeling the information store. This model is guided by both SGML and hypertext principles. The goal is to provide a conceptualization that allows documents to be contained in one or more files without loss of information and to allow for rich link (typed uni- and bi-directional) and anchor (point, region, and nodal) types. Major design goals included:

1. An extensible system of objects. Each object has a class, type, and subtype. At this point in time, folders and documents are the two major classes of objects defined, but other classes may be added. In addition, each class may be typed and subtyped. Thus folders are ordered and unordered at the type level and may be further subtyped. Documents types currently

include "CASCADE", "text", "SGML", "structured", "graphic", "image", "audio", etc. with a variety of subtypes as appropriate. Each class, type and subtype may have specialized server processing rules.

2. An extensible set of processing rules defining relations between objects. Thus, in an ordered folder, the sibling order of folders and documents may be defined. An ordered folder is essential to building a component SGML document. A document of type CASCADE can be a parent of other files -- allowing a more natural mapping of comment files.
3. An expanded set of access rules. Documents and folders in CASCADE may be protected at a number of levels with access provided to groups as well as projects. These principles are further defined below.
4. An extensible set of anchors and links. While the various types of anchors and links have not been fully defined at this point in time, comments have been defined as point anchors in the source document with the anchor label being an attribute of the anchor. Thus to the extent that comments take up no space in a document, it allows multiple users to comment using shared document locking. (Obviously, for editing, the document component has to be locked exclusively.)

What information will be maintained by the server?

The server maintains all information that is to be used by more than one individual. All information accessed by the group is maintained on the server. This includes project and group protections, document or folder class, type, subtype, id, location, description, and owner. The server also keeps track of creation, last access, last comment, and last modification dates, lock status, number of accesses, number of minutes accessed, and number of modifications.

While the storage location for most data is clear, there are some ambiguous situations. For example, we wish to record whether or not a user has seen a comment or a document. This could be stored on the client -- it is personal information -- but then it would not be available to other users -- has user x seen the comment made by person y. It could be stored in the DBMS, but the storage, processing and transmission overhead for hundreds of users would be high.

How will server information be stored (DBMS or file system)?

The design of the information store can vary from storing the entire document in an object oriented database to using the existing file system. We chose a hybrid approach. Files are maintained in a normal file system while access to the file system and control and meta information is maintained using a DBMS. This allows us to avoid the overhead and slowness experienced in many of the systems that stored document components in OODBMS, while offering the control advantages gained with the judicious storage of meta information in an RDBMS.

How will access to the information be controlled?

The server controls all access to documents and folders. Each document or folder is assigned a protection when

created (by default it is inherited from its parent.) The protection is a five digit code where each digit specifies a level of protection for a specific of action. Each user belongs to one or more groups that have access rights. Access is granted based on the ability of a user to provide proof that they have access rights greater than or equal to the protection level specified.

The access protections defined are applicable to the groups to whom access is granted. Thus being a member of a group with high access rights does not grant an individual access to a document unless the group is explicitly granted access. The author is always be assumed to have the highest access right. Any protection category for a file may be opened up to the project in which case actions at that level and below are open to all project members. The system is flexible in allowing changes in either user rights or document protections. It also allows flexibility in terms of creating specially restricted or privileged groups of people. As of this writing, we have failed to find any kind of reasonable document protection or access that cannot be provided under this scheme. At the same time, it is clear that arbitrarily complex access and protection schemes are allowed for under the system.

What processing will the server be responsible for?

There are many ways to split work between clients and server. In general, the server process all information related to the store and the work rules and clients provide for the interface. The situation is complicated when one considers low speed network connections. If data processed on the server expands it may be more efficient to transmit the compact data form and process it on the client. The converse will also be true, data that would most naturally be processed on the client may be processed on the server to reduce the number of data transfers or the size of the data transfer.

Once a decision has been made to process data on the server, the most appropriate method of processing the data must be defined. For example, in constructing a browser view in CASCADE, the current method is to begin with a given document ID and query the server for the children of that document recursively until a tree of the required depth has been built. The cost of these many data transfers, and the time required to build the tree time and again suggest that even though we violate principles of normalization, it may be necessary to keep a representation of the document tree on the server. This representation can simply be cut and pruned as necessary to respond to server requests.

In general, the server processes all data that needs to be shared between users and that benefits from the management capability provided by the DBMS. This includes records of all document activity, all the documents themselves, information about macro level user activity -- e.g. who is currently connected, and project semantics.

CONCLUSIONS

The CASCADE prototype demonstrates the feasibility of a

network centric environment for collaborative authoring. The proposed architecture and interactions can facilitate group activities. The system has produced a model for the document store that appears scalable and robust. In general, the extended protocols work well in both local area networks and wide area networks. The system shows promise of easing the navigational process by using visual techniques. The process of collaboration has been aided by augmentation techniques and the use of auxiliary information streams.

This research is still in its early stages. A number of questions are still being explored. Among these are the development of an optimized application protocol and refinement of the database structure. There are also questions about overall process efficiency that may require violation of some of the conceptual principles of DBMS design. While locking has been implemented at a gross level in the current version, a more accurate assessment of the capability of the system will have to await the implementation of element level locking which is currently being implemented. Also, the incorporation of augmented XML document construction and XML document display will represent an important step in the refinement of the system as a usable system for practical application.

ACKNOWLEDGMENTS

This work is supported in part by a grant from the Information Technology Laboratory of the National Institute of Standards and Technology, www.nist.gov/itl. In particular, we would like to thank Shukri Wakid, Sharon, Laskowski, and Charles Sheppard for the help and support.

REFERENCES

1. Bentley, R., et al., Basic Support for Cooperative Work on the World Wide Web, *International Journal of Human Computer Studies* 46, 6 (1997), 827-846.
2. Bentley, R., et al., Supporting collaborative information sharing with the World Wide Web: The BSCW shared workspace system, in *Proc. World Wide Web Journal, 4th International WWW Conference*. (December 1995, Boston), 1995, pp. 63-73.
3. Bentley, R., et al., Architectural Support for Cooperative Multiuser Interfaces, *Computer* 27, 5 (1994), 28--36.
4. Berners-Lee, T., The World Wide Web: Past Present and Future. W3C, <http://www.w3.org/People/Berners-Lee-Bio.html/1996/ppf.html>, 1996.
5. Brown, J. S. and P. Duguid, Borderline issues: social and material aspects of Design, *Journal of Human-Computer Interaction* 9, 1 (1994), 3-36.
6. Ede, L. and A. Lunsford, *Singular Text/Plural Authors: Perspectives on Collaborative Authoring*. Southern Illinois University Press, Carbondale, 1990.
7. Eklundh, K., Problems in Achieving a Global Perspective in Computer-Based Writing, in *Computers and Writing: Issues and Implementations*., Kluwer Academic Publishers, Dordrecht, the Netherlands, 1992.
8. Ellis, C. A., S. J. Gibbs and G. L. Rein, Groupware: Some Issues and Experience, *Communications of ACM* 34, 1 (1991), 39-58.
9. Engelbart, D. C., Augmenting Human Intellect: A Conceptual Framework. Stanford Research Institute, Technical Report AF 49(638)-1024, 1962.
10. Engelbart, D. C., *Vistas in Information Handling*. Spartan Books, Washington, DC, 1963.
11. Engelbart, D. C., Authorship Provisions in AUGMENT, in *Proc. Proceedings of the COMPCON Conference*. (February - March 1984,), 1984.
12. Engelbart, D. C., Augmentation Research Center, in *A History of Personal Workstations*., Addison-Wesley Publishing Company, New York, New York, 1988, pp. 187--232.
13. Engelbart, D. C. and K. Hooper, The Augmentation System Framework, in *Interactive Multimedia*., Microsoft Press, 1988, pp. 14--31.
14. Ferguson, I. A., Touring Machines: Autonomous Agents with Attitudes, *Computer* 25, 5 (1992), 51--55.
15. Fish, S., et al., Quilt: A Collaborative Tool for Cooperative Writing, in *Proceedings of COIS'88 Conference on Office Automation Systems*., ACM SIGOS, Austin, TX, 1988.
16. Haake, J. and B. Wilson, Supporting Collaborative Writing of Hyperdocuments in SEPIA, in *Proceedings the ACM Conference on Computer-Supported Cooperative Work (CSCW '92)*., 1992.
17. Halasz, F. and M. Schwartz, The Dexter Hypertext Model, *Communications of the ACM* 37, 2 (1994), 30-38.
18. Hardman, L., D. C. Bulterman and G. V. Rossum, The Amsterdam Hypermedia Model, *Communications of the ACM* 37, 2 (1994), 50--62.
19. Hearst, M. A., TileBars: Visualization of Term Distribution Information in Full Text Information Access, in *Proceedings of Human Factors in Computing Systems, CHI'95*., ACM, Denver, CO, 1995, pp. 59--66.
20. Heo, M., et al., Multi-Level Navigation of a Document Space, in *Proc. Proceedings of WebNet'96*. (October 1996, San Francisco, CA), AACE, 1996.
21. INRIA, Alliance: a web based structured cooperative editor., 1996.
22. Kay, A., User Interface: A Personal View, in *The Art of Human Computer Interface Design*., B. Laurel, Ed., Addison-Wesley Publishing Company, Reading Massachusetts, 1990, pp. 191-208.
23. Kumar, H., C. Plaisant and B. Shneiderman, Browsing Hierarchical Data with Multi-Level Dynamic Queries and Pruning. Dept. of Computer Science at Univ. of Maryland, Technical Report CS-TR-3474, 1995.
24. Licklider, J. C. R., Man-Computer Symbiosis, in *Proc. IRE Transactions on Human Factors in Electronics*. (March 1960,), 1960.
25. Mandivwalla, M. and L. Olfman, What do Groups Need? A Proposed Set of Generic Groupware Requirements, *ACM Transactions on Computer-Human Interaction* 1, 3 (1994), 245--268.
26. Maurer, H., *Hyperwave: the Next Generation Web Solution*. Addison-Wesley, 1996.
27. Morse, E. L. and M. B. Spring, Visualizations that Support Groupwork, in *Proc. IEEE Symposium on Visual Languages*. (September 1996,), IEEE Ablex, 1996.
28. NCSA, NCSA Habanero Project., 1996.

29. Neuwirth, C., et al., Computer support for distributed collaborative writing: defining parameters of interaction, in *Proceedings of CSCW94.*, ACM, Chapel Hill, NC, 1994.
30. North, C., B. Shneiderman and C. Plaisant, User Controlled Overviews of an Image Library: A Case Study of the Visible Human. Dept. of Computer Science at Univ. of Maryland, Technical Report CS-TR-3550, 1995.
31. Olsen, K. A., et al., Visualization of a Document Collection: The VIBE System, *Information Processing and Management* 29, 1 (1993), 69--81.
32. Plaisant, C., D. Carr and B. Shneiderman, Image Browsers: Taxonomy, Guidelines, and Informal Specifications Technical Report CS-TR-3282. Dept. of Computer Science at Univ. of Maryland,, 1994.
33. Rimmershaw, R., Collaborative Writing Practices and Writing Support Technologies, in *Computers and Writing: Issues and Implementations.*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1992, pp. 15--28.
34. Roseman, M. and S. Greenberg, Designing real-time groupware with GroupKit, a groupware toolkit, *ACM Transactions on CHI* 3, 1 (1996), 66--106.
35. Sharples, M., et al., Research Issues in the Study of Computer Supported Collaborative Writing, in *Computer Supported Collaborative Writing.*, Springer-Verlag, London, 1993, pp. 9-28.
36. Sharples, M. and L. Pemberton, Representing Writing: External Representations and the Writing Process, in *Computers and Writing: State of the Art.*, Kluwer Academic Publishers, Oxford, England, 1992, pp. 319--336.
37. Shneiderman, B., Tree Visualization with Tree-Maps: A 2-d Space-Filling Approach, *ACM Transactions on Graphics* 11, 1 (1992), 92--99.
38. Shneiderman, B., The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. Dept. of Computer Science at Univ. of Maryland, Technical Report CS-TR-3665, 1996.
39. Spring, M. B. and M. C. Jennings, Virtual Reality and Abstract Data: Virtualizing Information., *Virtual Reality World* 1, 1 (1993), C--M.
40. Spring, M. B., et al., Embodying Social Capital Facilitators in a Collaborative Authoring System, in *Proc. Association for Information Systems 1997 Americas Confernece.* (August 15-17, 1997, Indianapolis Indiana), Association for Information Systems, 1997.
41. Spring, M. B., E. L. Morse and M. Heo, Multi-level Navigation of a Document Space, in *Proc. Proceeding of Leveraging Cyberspace Conference.* (October 1996, Palo Alto, CA), NIST, 1996.
42. Spring, M. B. and M. B. Weiss, Alternatives to Financing the Standards Development Process., in *Twenty-Second Annual Telecommunication Policy Research Conference.*, Solomons, MD, 1994.
43. Zuboff, S., *In the Age of the Smart Machine: The Future of Work and Power.* Basic Books, Inc., 1984.