On Computer Supported Collaborative Writing Tools for Distributed Environments

Technical Report 94-14

Kai. H. Chang, Yu Gong, Tim Dollar, Shefali Gajiwala, Byong Lee, and A. Wesley Wear

Department of Computer Science and Engineering
Auburn University
Auburn, AL 36849

Abstract

Traditional computerized writing tools designed for single user access often create barriers to group collaboration. Usually, participants have to work in an interleaved fashion to prevent inconsistency. Oral communication is normally used for coordinating group activities. With the recent development of computer networks and widespread deployment of networked workstations, automating the group writing process for geographically distributed users has become feasible. This paper introduces a Distributed Collaborative Writing Aid (DCWA) developed for networked workstations. The DCWA can help users cooperate on a writing task (such as programming, report writing, note taking, etc.) logically, conveniently, and efficiently. Among many important open problems, the paper addresses issues related to the four major components in the DCWA. These are group organization, multicasting within groups, distributed database, and user interface.

Key Words: Computer Supported Cooperative Work (CSCW), collaborative writing, multiuser environment, computer network, database systems.

1 Introduction

In a conventional problem solving environment, computers normally provide passive support to a group of users working as a team — while computers store, analyze, retrieve, and present data, the decision process governing the logical and orderly interactions among team members is left to the users. As the pressure of improving productivity continues to grow, new technologies must be developed to provide active support for cooperations and to overcome geographic barriers [1]. This demand leads to the recent research and development of *computer supported cooperative work* (CSCW) [2, 3, 4, 5]. The collection of hardware and software supporting CSCW and the like is also known as *groupware* [6] with a more technically oriented meaning.

Centralized groupware (i.e., either large mainframe or PC based software) was explored in the past as a means for achieving *office automation* [7]. Only recently, the focus has been shifted to the development of networked distributed groupware [7]. Many problems remain open in a distributed environment. The most salient ones are the *group organization*, *multicasting*, *distributed database management*, and the *user interface*. These problems are mutually dependent and cannot be dealt with in isolation. In this paper, we describe the experimental work toward a distributed groupware, known as *Distributed Collaborative Writing Aid* or DCWA. The DCWA is targeted at providing some specific solutions to the above problems in a restricted application domain, i.e., the collaborative writing.

As explained in [5], currently there are very few systems supporting collaborative writing. It is argued that it is still premature to provide solid guidelines for such system development. It is therefore suggested in [5] that "Don't rush to invent systems for collaborative writing until the process is better understood." Nevertheless, we believe experimental systems for collecting lessons/experiences is both worthwhile to develop and desperately needed in this new research arena. Moreover, some general requirements for collaborative writing have long been recognized in the literature [4, 8]. These include, e.g.,

- Provision of communication and exchange of information among authors.
- Mechanisms for sharing information to allow cooperation.
- Management of inputs and outputs for users.
- Mechanism for role control for the team members.

The DCWA includes most of the characteristics required of such tools identified by other researchers. It also includes many features we feel to be necessary for this type of software.

Specifically, the DCWA provides a *distributed asynchronous and synchronous* collaborating environment. Users initiate, join, or end collaboration through the system's *groups services* facility. A distributed *multimedia database* organizes both textual and graphical information according to their structural relationship as well as their semantical relationship. Users can manoeuvre in the document logically using a semantic network that is overlaid on the document structure. Finally, the user interface helps the user see his own focus and other group members' working areas with either WYSIWIS (What You See Is What I See) or WYSINWIS (What You See Is Not What I See) style. The WYSIWIS style is typical for synchronous cooperation where members must see changes instantly, while the WYSINWIS style is important in case information must be tailored before being presented to a remote user. A mixture of these two styles appears to be more useful.

The paper is organized as follows. In the next section, we provide a brief overview of the previous research results in this area. In Section 3, the design methodologies for the DCWA are described and contrasted with the previous work. Section 4 contains the concluding remarks and discussions of possible extensions.

2 CSCW Systems

Many systems supporting CSCW collect information from the participants and suggest the appropriate courses of actions to take. The SYBIL system [9] collects and coordinates options about a specific design topic from members of a hardware platform project team, and suggests them goals and subgoals. The Office Works system [10] arranges meeting for a group of participants by checking each participant's schedule.

CSCW can be accomplished through multimedia communication channels. The MERMAID [11] and the SPIN [12] systems are prototypes that provide real-time conferencing environments for geographically distributed participants by using synchronous textual, audio, and video communications. One of the drawbacks associated with these systems is that the cost is far beyond what most users can afford. A simplified version of conferencing service is to use extended textual bulletin boards [8]. In addition to providing conferencing service, the COGNOTER system [13] also collects and organizes ideas from its participants for discussion. It is one of the piloting systems that provides the WYSIWIS capability.

A fundamental requirement for CSCW is the provision of information/message exchange among the participants. It has been suggested that messages should be organized in a structured way to achieve efficiency [14]. That is, protocols must be designed to differentiate cooperative information from those of little relevance, e.g., the cognitive filtering in Information Lens [15] and the semi-automatic agents in Object Lens [16]. The second requirement is the provision of mechanisms for information sharing and cooperation [8]. Most CSCW environments stress the need for the distinction of public and private information. However, as described in [17], information to be used in cooperative work needs to be taken out the limit of "personal wall." It was suggested that all users should have common access to all information.

In the field of collaborative writing, the collaboration among coauthors can be classified into two categories: *shared mind* and *division of labor* [5]. In the first category, multiple authors edit the same file at their own locations and a coordinator integrates the contributions. In the latter category, a task is divided into mutually independent subtasks for coauthors. A merging section then takes place to integrate the subtasks. These classifications are parallel to the asynchronous and the synchronous collaboration. In the asynchronous approach, users do not work at the same time; while in the synchronous approach, users work at the same time on the same file.

An experimental system providing asynchronous collaborative writing for a small group has been reported in [3]. The system uses email as the communication media and the text is represented in the ASCII format. The advantage of this approach is that the impact of the geographical locations and the computer variation among the team members is kept minimum. However, since only email is used, referencing comments or changes to existing files may be difficult. An interesting observation made by the developer was that the success of collaborative writing depends heavily on the mutual confidence and trust among the team members.

A hypertext system provides comprehensive services for document writing and reviewing and is normally distinguished by two features — links and windows [18]. A document is organized into nonlinearly linked nodes. A link may indicate a referential or organizational relationship. A window is used to provide the display and editing services of the content of a node. It also allows a user to traverse the links through the window service. The feature of linked node organization is not only limited to the display, there must be a one-to-one node correspondence in the database too. Although hypertext provides new possibilities for authoring and design, it suffers from the disadvantages of user disorientation and cognitive overhead. The technology is mainly for a single user working alone at any given time. Quilt [19] is a hypertext system that provides co-authoring services. The users are identified as either coauthors or commenters. It allows all users to comment on a document. However, only certain privileged users, i.e., the coauthors, can modify the document. The ForComment system [10] provides a group editing environment. It allows up to 16 reviewers to comment on a document. However, only the original author can actually modify the document.

The rIBIS [20] and the SEPIA [21] are synchronous hypertext systems providing various levels of collaboration modes. In the independent mode, users may work on their own tasks without interfering each other. In the loosely-coupled mode, users may share certain public information while working on their own tasks. In the tightly-coupled mode, users will share the same view, and resources, e.g., mouse and file, are strictly controlled to avoid conflicts. The major improvement of SEPIA over rIBIS includes the automatic mode switching and the use of composite nodes. SEPIA also provides audio communication channel between participants.

The DCWA presented in this paper addresses many issues identified in the literature. It can be utilized in either asynchronous or synchronous mode. A team member may double click on a graphical representation of another user's working area to pop up a window for its contents. Once the window is popped up, change of its contents is synchronous with the actual modifications that happen on a remote site. The tool is also asynchronous in the sense that all users may freely quit and rejoin the group at any time in the writing process. In addition to the text editor of most conventional collaborative writing systems, the DCWA also provides a graphical editor.

The composite node concept of [21] has been adopted to organize the file structure. A tree-like hierarchical structure is used to define the non-overlapping working spaces among users. This structure normally reflects the organizational structure for the document, such as the separation of the document into chapters, sections, subsections, etc., and does not carry much information about the contents. Therefore, in the DCWA, a semantic network can be specified by users and overlaid upon the organizational structure to allow more meaningful migration from one node to another. The semantic network also serves for another purpose: it allows information in one node to be shared from another node. For example, the score of a baseball game may be entered only once in a node and be used from many other nodes. Clearly, the organizational structure follows the "division of labor" principle, and the provision of the semantic network follows the "shared mind" idea. These two ways of collaborations are combined in the DCWA.

3 The DCWA Design Methodologies

The DCWA is a distributed textual and graphical integrated editing tool implemented in many standard programming facilities, including 4.3BSD Unix, the TCP/IP protocols, the OSF's Motif toolkits set, the ANSI C, and C++. Portability is, therefore, automatically maintained. Among many objectives, the following are the major goals for the design of this software.

- *Providing group services.* In the development of the DCWA, *group* is the first most important concept that must be supported so that every member can contribute to the right document at the right place.
- *Maintaining a unique version of the document.* Even though members of the same group are scattered geographically, they must work on one unique version of the file all the time.
- *Facilitating both organizational and semantical relations among parts of the document.* This objective is motivated from combining "division of labor" and "shared mind" working styles into a hybrid style, as discussed in the previous section.
- *Providing WYSIWIS interface for the current working area.* This capability allows users to see what others are doing instantly.
- *Providing WYSINWIS style interface for users' own viewing spaces.* Each user can define his/her own logical *view* of the overall document space based on his/her own interest and responsibility. (Thus, "section 2" for user A may be "section 3.2" for user B and "section 4" for the entire group.) For a large document, this provision would be a useful feature.

In the remainder of this section, we discuss our design methodologies for achieving these objectives.

3.1 The Software Architecture

The overall organization of the software is based on an object-oriented approach. Figure 1(a) shows the four objects on each user host. The primary functions of each object are explained below. In the subsequent three subsections, the details of the Group Server, the Database, and the Graphical User Interface (GUI) are further explained.

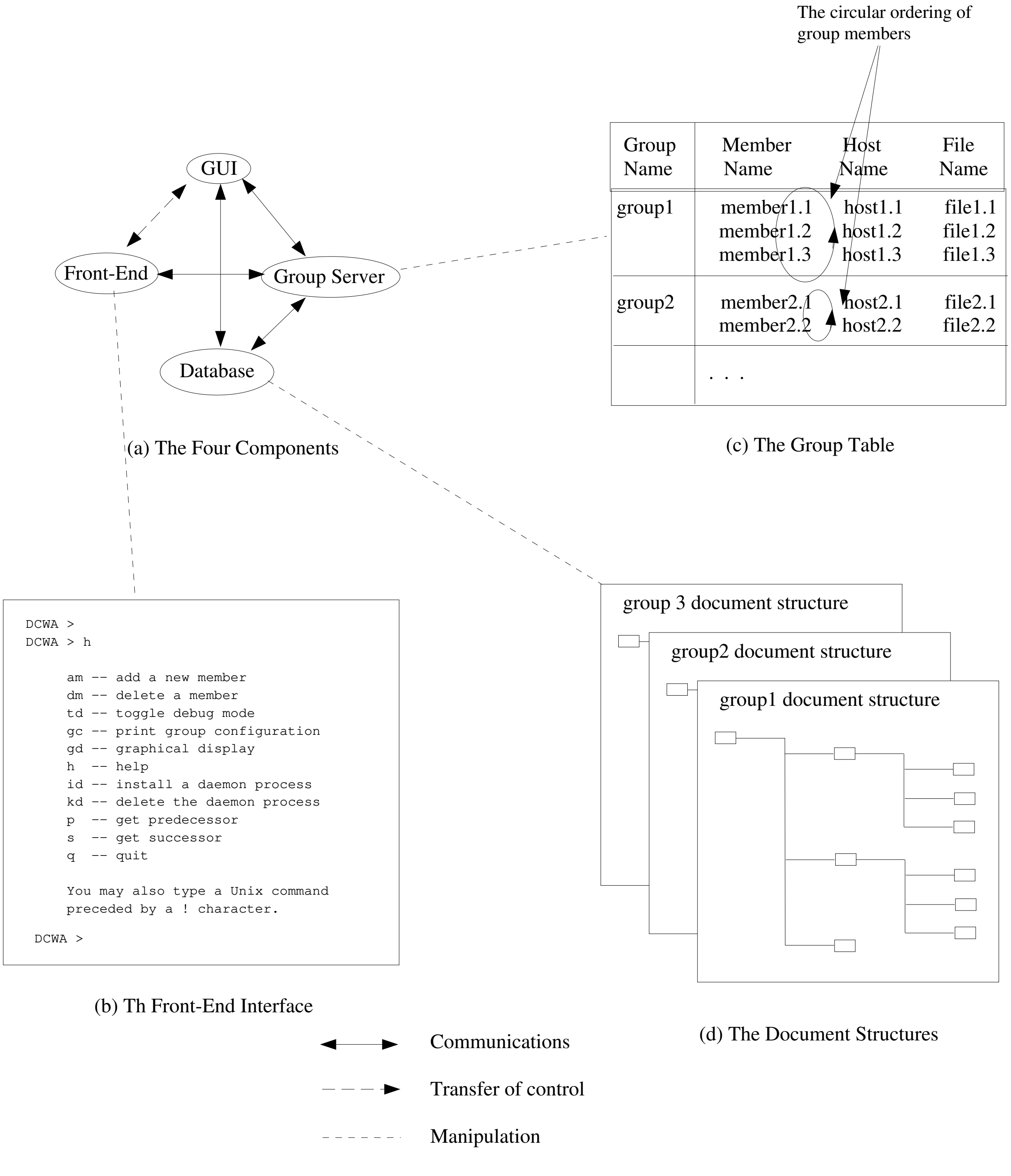

Figure 1: The DCWA Architecture

- *The Front-End Object.* This is the first object that a user interacts with. Two options are provided: text-based interaction and graphical interface. In case the user selects the graphics option on the command-line, the Front-End is automatically skipped and the user directly interacts with the GUI. The major purpose of the text-based Front-End is for the user to interact with the Group Server (for creating a group or for joining an existing group, and for registering some necessary information) before invoking the full-scale graphical interface. It is useful for a first-time user who is still not a member of any group. In such a case, the GUI, if invoked, would not show any contents. The text-based Front-End and the available commands is shown in Figure 1(b).

- *The Group Server Object.* The Group Server maintains a *group table* as shown in Figure 1(c). All members in a group are assumed to have unique member names. They are circularly ordered as shown in Figure 1(c), i.e., each member has a unique predecessor and a unique successor. Furthermore, a document is distributed with every member possibly holding one piece. The column labelled with "File Name" in Figure 1(c) shows the file names[1] for all pieces of a document on individual member machines.

- *The Database Object.* How the pieces of a document fit together is managed by the Database Object. It maintains an organizational structure of the document as shown in Figure 1(d). This is a tree structure as explained in Section 2. The Database keeps the mapping between a node and its owner's member name. Since all members in a group have unique names, by checking with Group Server, all pieces of the document can be retrieved. In addition, members can define his/her own view of the document to be consistent with the global organizational structure.

- *The Graphical User Interface Object.* The GUI object may interact with all other objects and its peer GUI processes in case timely update of window contents is necessary.

We believe except for the Front-End object, the other three objects are the minimum functional components that any distributed CSCW writing tool must provide. In the following subsections, we discuss the details of each of these three objects.

[1]In fact, the *file paths* are recorded in the Group Server. The files can also be retrieved using the http on the Internet in case the Group Server is not running on a remote machine.

3.2 Group Services

The major challenge in the design of the Group Service comes from the communication complexity, rather than the complexity for managing the group data structure on a host.

The major difficulty is the multicasting of messages to all members in a group. It is found that occurrences of almost every event has to be made known to all members in the group. Experiments show that it is infeasible to send duplicated messages to all member machines. We measured that in a local area network (LAN), the delay between a pair of machines is between 5 *ms* and 100 *ms* when a single message is sent. When more messages are sent, the average delay appears to increase exponentially. In such an environment, it is beneficial to reduce the amount of messages passed over the network. Therefore, in the DCWA, a multicast message is not duplicated for multiple destinations; instead, all members of a group are arranged in a circular ring as shown in Figure 1(c), and only one message is sent over the ring. This message would eventually pass through all members. When it completes a circle, the original host removes the message. Our experiment shows that the delay is satisfactorily short. This result is expected since, e.g., the cycle time over a ring of 10 users is at most 1 second in a LAN. Note that the DCWA are supposed to provide service to humans. A one second delay is virtually unnoticeable.

Here we must point out that our observations are valid for LANs, not for switched wide area networks (WANs). We believe to provide synchronous long-haul collaboration services, some fundamental innovations are needed in the current WAN technology. This problem is beyond the scope of this paper.

3.3 The Distributed Database

The Database object is crucial in making the entire system work logically and consistently. Many issues are typical for all distributed databases such as backup and recovery, defining views, and maintaining logical relationship among entities (here called *nodes*). Some other issues are unique to our system, notably the overlay of a semantic network over the document organizational structure.

Backup and Recovery

The database process provides full service for recording local and remote users' work areas and the organizational structures of the document. Rollback, as is a common capability in all commercial transaction systems, is supported. A major characteristic of our transaction system is that rollback is rarely meaningful only to one or a few users. Oftentimes, it must be multicast to *all* group members as quickly as possible. Because of this property for collaborative writing applications, cascaded rollback, as is the major challenge in some event-driven applications such as the Time Warp system for discrete-event simulation [22], would not be a problem in our database system, which is also an event-driven system.

Distributed Storage of Single-Version Documents

In the DCWA, files belonging to a group can be geographically scattered. It is completely unnecessary that all files for a group be located in the same file server. The organizational structure mentioned earlier provides relations among all parts of the document, called *nodes*. To maintain a single version of the document, all nodes are assigned unique owners, which are the users who created the nodes. From time to time, files owned by one user may have to be loaded down to other machines, but all modifications must always be committed in the master copies stored at the owner sites. This is slightly similar to the SYBIL [9] and Office Work [10] where a central server collects different and possibly conflicting versions from members and try to resolve any conflict. Different from them, the DCWA does not broadcast the resolution back to members since all members in the group are able to perceive changes made by others through the multicast messages, and then do conflict resolution (if it is necessary) locally.

Semantic Network

Many existing CSCW tools only allow users to associate a label with each node in the document's organizational structure for fast migration from one part to another. We observe that information that can be carried by one label is quite limited. We, therefore, carry on the idea of node-labelling to its full extent: mechanisms are designed and implemented for users to assign *attribute-value pairs* to any node. A search facility with an SQL-style interface is provided for users to look for all nodes that satisfy a specified condition.

For example, in an academic writing application, a node may correspond to a segment of text that mentions a certain chemical compound c_1. For future reference, the user may assign the following attribute-value pair to the corresponding node.

$$(\text{CHEMICAL\_COMPOUND}, c_1)$$

Here we have used a word in all capital letters to indicate that it is an attribute name.

In addition to assigning attribute-value pairs to a node, the database also helps the user to relate all attributes in an *inheritance* structure. For example, the user may find that "CHEMICAL_COMPOUND" is a subconcept of "CHEMICAL." Therefore, the user may add the following entry in the database which is taken to be globally valid.

(CHEMICAL, CHEMICAL_COMPOUND)

Clearly, the attribute-value pair can be used to encode the "is-a" semantic, and the attribute-attribute pair can be used for encoding the domain relationship among concepts. These two types of semantics are believed to be the fundamental elements for all possible semantic relations [23]. In the design of the DCWA, they are implemented using a monolithic mechanism. We will call it the method of "attribute-value pairs" in the sequel.

By associating each node with the attribute-value pairs, a semantic network is overlaid upon the organizational tree. An example (the academic writing example) is shown in Figure 2. Using the semantic network and the search facility, users may compose queries to locate all nodes that satisfy certain condition, e.g., "All nodes that mention the chemical c_1 and mention something about chemical reaction." The result of the query may be used for finer search until the needed information is found, or it may be used to define a view (see below) for the user to focus his/her attention. This functionality is similar to what is provided by the Information Lens [15] and the Object Lens [16]. It is potentially important for new members to define his/her work scope.

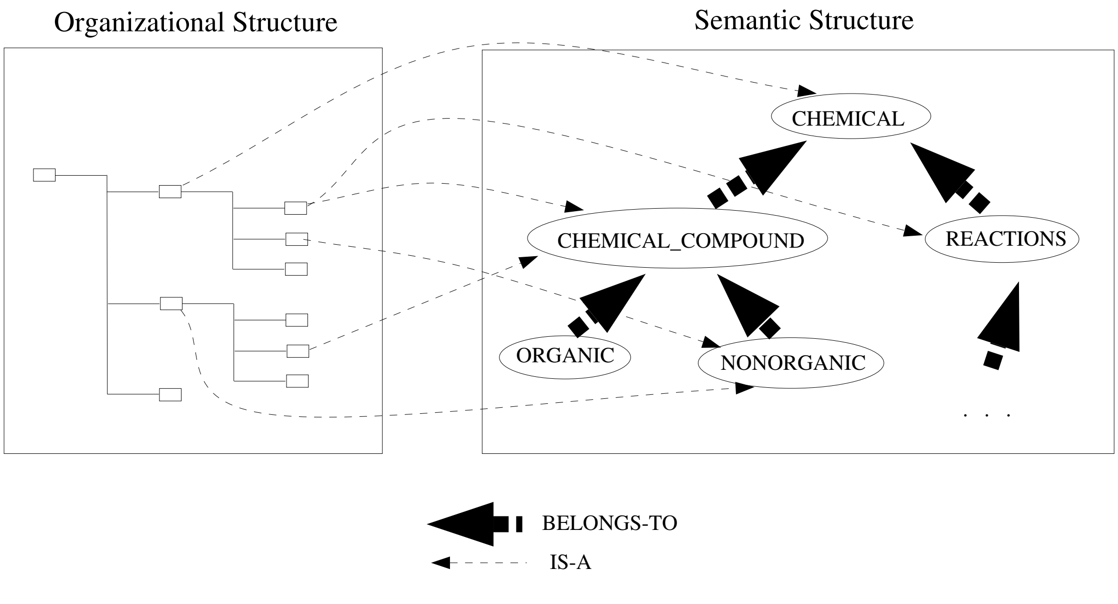

Figure 2: An Example Semantic Network

Finally, it is interesting to note that the hypertext solution in Quilt [19], ForComment [10], rIBIS [20], and SEPIA [21] is, in fact, a primitive form of overlaying a semantic net over the document structure. Unfortunately, hypertext links normally lack good organizations. Users may easily get confused. The hypertext organization also lacks a similar theoretical foundation enjoyed by the semantic networks. However, the hypertext solution is clearly more efficient than the general semantic network approach. We expect that for writing component-wise self-contained document, such as a users' manual, a tool based on hypertext would be more advantageous. On the other hand, for writing a document that draws facts from various parts of its own, such as a technical paper or a large program, a more powerful semantics mechanism is needed.

Definition of Views

The DCWA makes the entire document available to all group members at any time following the principle of "no personal walls" [17]. In reality, however, not all users are interested in the entire document until possibly the final stage of the writing process. Therefore, users are provided mechanisms to select only those nodes that he/she is most interested in. This provision is, of course, not difficult to implement once the mechanism for manipulating the logical structures have been implemented, but we feel it is important for us to point out the necessity for private views of the document in a collaborative environment where the document may potentially grow extremely large.

Sharing and Conflict Resolution

Sharing in the this prototype is handled manually by users through locking and unlocking nodes in the logical structure. A node is locked automatically when it is selected by a user, and unlocked if the user selects another node. Users may voluntarily unlock a node if he/she feels necessary. Conceivably, users must discuss among themselves about who should monopoly a given node. Possibly in the future version of this system, complete sharing and automatic conflict resolution will be implemented which allows users to work without knowing the whereabout of others. For this version, sophisticated conflict resolution algorithm is not within our main focus. Thus, a trivial (but working) approach has been taken.

There have been a number of real-time conferencing tools that include conflict resolutions, such as MERMAID [11], SPIN [12], and the textual bulletin board in [8]. We found that their methods are not quite effective for resolving conflict arising in document composition. The ideal method for resolving conflict of two or more writers is by *combining* their contributions into a consistent piece, while in the conference environment, "knock out" is normally an acceptable solution.

3.4 Graphical User Interface

The GUI at each local machine provides a user with the necessary functionality to participate in the cooperative work. In a complicated system like the DCWA, the amount of information pertinent to the work status of a group can be overwhelming. It is important that the GUI provides the desirable information at appropriate time. The design of this GUI is based on the premise that, unless requested or immediate attention is required, information is hidden from the user. This just-in-time philosophy is designed to reduce the mental burden on the user. When the GUI is initiated, only a basic text editing window with various option icons is provided. A user can choose to work alone or work with other group members by tailoring the interface layout (using the mouse) to meet his/her own need. The services provided by the GUI can be described in five categories: status information, search tools, text and graphics manipulation and integration, and user coordination.

Status Information

The GUI provides cooperation information ranging from application groups to local file structures. At the group level, the application groups, their members, and the working status of each member are available. At the file structure level, the structure of a selected file is displayed along with the color coded nodes to indicate which nodes are currently edited by other users and, therefore, not available. In Figure 3, the file structure of a graphics program along with the content of a selected node "Create-Menu" are shown. From the figure, it can be seen that the node "Create-Menu" has been changed to a darker color. The bottom portion of the window contains the messages from the GUI.

A user can also request to view any member's working window, provided that they belong to the same group. As shown in Figure 3, two other members' working windows are also displayed. This information viewing capability makes the interaction and discussion among users more productive.

Search Tools

The document that a group of members are creating can be complicated and is represented in a file structure which resembles a tree hierarchy. In order to help a user to locate a desired portion of the document quickly, a search facility is provided.

In addition to the actual node content, each node in the file structure also contains attributes that describe the nature of the node. Typical descriptions can be a series of keywords or concepts of the node content. A search tool that allows a user to specify the node descriptions can locate a desired node quickly. In this system, the node specification can be any logical combinations of the node attributes and the located nodes are colored coded in the structure for easy identification. In the example of Figure 3, node search can be accomplished by utilizing the "Node-Search" pull-down menu. This node labeling/indexing and search technique distinguishes DCWA from other CSCW systems.

Text and Graphics Manipulation

A user can select a working area from the file structure by locking a leaf node. The content of the node will then be displayed in an editing window. The GUI provides both textual and graphical editing capabilities, which are available in separate windows. The textual editor utilizes the Motif text editor while the graphic editor is implemented in C++.

The user can create, modify, and update the content of a node. Splitting and merging nodes and restructuring of the file structure are also allowed. Any updates performed in an editing window will be sent to other group members on a real-time base for information sharing. This real-time viewing window provides the WYSIWIS capability [13] to the user.

When a user selects a node to work on, the content of the node is retrieved and stored at the user's local site. At the end of the editing session, the revised content of the node is passed to the database process for storage. If the user tries to exit the node without saving the contents, a warning prompt will be signaled for confirmation.

Text and Graphics Integration

The graphic editor provides a special viewing capability that multiple graphs can be placed on top of each other as an integrated drawing. This capability is essential because when multiple users are designing different aspects of an object, the relativity of various designs should be maintained. For example, a house design may contain floor, plumbing, electrical, and air-conditioning subplans. Different members of a design team may have to design their subplans based on the main subplan, in this example, the floor plan. A member can achieve this by placing the main drawing in the background and using it as the reference for his own drawing. He is also able to display other members' drawings in the same window and make sure that all the subplans fit together. Since most subplans make reference to the main subplan, whenever there is a change to the main subplan, the other subplans would be changed accordingly. However, if the changes to the main subplan is too drastic, like deleting a room, the automatic adjustment will not take place, instead the designers would have to manually perform the changes.

The content of each graphical node is stored separately just like the textual contents. However, unlike the textual contents, graphical contents of different nodes can be integrated or overlaid and, then, used to form a new node. Figure 4 shows a house design that contains both the floor subplan and the air condition ducking subplan. In addition, textual and graphical contents can be integrated. This is to be accomplished by transforming the graphical contents into the postscript form and then importing it into the textual portion.

User Coordination

The coordination among multiple users are achieved through a loosely coupled fashion. The tightly coupled coordination mode of [20] is not needed in this textual and graphical editing environment. The GUI provides several types of coordination services to avoid conflicts among users. At the global level, application groups can be created and the group creator can decide who is allowed in the group. Each group maintains its own membership and file directories. Group members may work on the same file and access other members' views. As described earlier, a user can only select a leaf node that is not currently selected by other users to work on. When a node is selected, its color on the file structure will change to indicate that it is unavailable to other users. By using this locking mechanism, information consistency can be maintained.

If a user needs to view other group member's current work, he may simply choose the desired node from the file structure. The current content of the desired node will be displayed and updated in a separate window on a regular basis (currently set at every 15 seconds). This viewing window is displayed upon request only and a user may choose to view as many team members' work as desired. In the example of Figure 3, two other members' windows are shown. This viewing capability coupled with a "talk" window (see below) provides group members a mechanism to discuss and make reference to the file contents.

The GUI also furnishes a talk window that allows a member to communicate with other members. A member can select from the group member list for interaction partners. If the other member does not have the talk window opened when a talk request arrives, a request confirmation message would be signaled. Group members may utilize this channel to discuss topics related to the file contents and structure, application group arrangement, and others. The window contains two parts, the first part displays messages from other members while the second part allows a member to type messages. In order to provide a feeling of real-time interaction, the message is passed to other members in the fashion of character by character.

4 Discussion and Conclusion

In this paper, a distributed cooperative writing tool, DCWA, is presented. It provides both textual and graphic group editing, viewing, and coordination services to team members. The design of the tool follows the spirit of "division of labor" through the file structure locking mechanism. However, the "shared mind" approach can also be realized by viewing other members' work spaces. The flexibility of the tool also allows it to be used in either synchronous or asynchronous mode.

Our experience of using DCWA to write programs indicates that the capability of viewing other team members' current work are extremely beneficial to all members. Because the way the viewing is set up, a member is not bothered when other members are viewing his/her work. Unless absolutely necessary, a member's work would not be interrupted. When an interactive communication is needed, the talk window would provide the service. It is interesting to note that, many times, a user chooses not to respond a message in the talk window immediately for various reasons. However, when an interactive talk is involved, the viewing windows provides very helpful references.

The experience also confirms the finding of [3] that the success of collaborative writing depends on the mutual confidence and trust among the team members. Since members of the same application group have equal access rights, e.g., viewing and modifying, to all information belonging to the group, members must be willing to disclose contribution to each other. Because of the small size of our research team and the member responsible for each node in the file structure is understood among the team, this no-privacy nature does not pose any difficulty for us. Actually, no one likes to perform editing beyond his/her assigned nodes. However, sometimes a member must modify other member's node to make the complete program consistent. For example, a modification to certain nodes may be needed to make the program compile, but the member responsible for the nodes may not be notified for the changes after the fact. In order to reduce the unintended miscommunication, an editing log for each node will be established to record the last few, e.g., five, editing users. When a member notices other member has modified certain nodes, he/she may want to find out what has been changed.

When a team is formed to write a program, an anticipated file structure is normally derived in a group discussion and then created in DCWA. This structure provides the initial division of labor style assignment for the team. However, as the writing progresses, members tend to reorganize the structure quite often by moving a node to another branch. In our programming experience, the graphical editing capability was not fully utilized, but members use the graphical editing window for real-time intensive interactive discussion. A temporary graphic node is created and deleted in the file structure just for this purpose.

The node search tool has been very useful in helping the team members locate a desired node, e.g., a subroutine or header segment in a program. This is particularly true when a member tries to find out the communication protocols among processes. Many times a member would like to see the actual definition of a protocol in the program before he/she is satisfied. Through appropriately defining node attributes-value pairs, a user can easily migrate from one node to another and then back to the original location. Without such a facility, browsing through all the nodes to search for the desired code becomes the only means. In our example of approximately 12,000 lines of C and C++ code, the search process would be tedious.

The work reported in this paper is still ongoing. The integration of the graphical and the textual contents will be accomplished in a near future. The editing log of each node, the data compression for the graphical contents, and some detailed node search functionality are still to be completed. Evaluation of the environment by different application groups, such as project report preparation, team-taught course materials preparation, and user manual writing, will provide more insight about the adequacy of the design. The experience gained from the design and implementation of this version of DCWA will no doubt contribute to the future development.

References

[1] J. Cuena and A. Garcia-Serrano. "Intelligent Computer Support". *Cooperation among Organizations — The Potential of Computer Supported Cooperative Work*, pages 72–102, Power, R.J.D. (Ed.), Germany: Springer-Verlag, 1993.

[2] R. J. D. (Ed.) Power. *Cooperation among Organizations — The Potential of Computer Supported Cooperative Work.* Germany: Springer-Verlag, 1993.

[3] D. Diaper. "Small-Scale Collaborative Writing Using Electronic Mail". *CSCW in Practice: An Introduction and Case Studies*, pages 72–102, Diaper and C. Sanger (Eds.), Germany: Springer-Verlag, 1993.

[4] B. Hewitt and G. N. Gilbert. "Groupware Interface". *CSCW in Practice: An Introduction and Case Studies*, pages 31–38, Diaper and C. Sanger (Eds.), Germany: Springer-Verlag, 1993.

[5] M. Sharples. "Adding a Little Structure to Collaborative Writing". *CSCW in Practice: An Introduction and Case Studies*, pages 51–68, Diaper and C. Sanger (Eds.), Germany: Springer-Verlag, 1993.

[6] J. N. Orr. "Graphics and Groupware: Increasing Intimacy through Broadening Bandwidth". In *Groupware'92*, pages 73–76, Morgan Kaufmann Publishers, 1992.

[7] J. Grudin. "Computer-Supported Cooperative Work: History and Focus". *IEEE Computer*, pages 19–26, May 1994.

[8] T. Rodden. "Technological Support for Cooperation". *CSCW in Practice: An Introduction and Case Studies*, pages 1–22, 1993.

[9] J. Lee. "SIBYL: A Tool for Managing Design Rationale". In *Proc. of the Conference on the Computer Support Cooperative Work*, Los Angeles, CA, 1990.

[10] S. Opper. "A Groupware Tool Box". *BYTE*, pages 275–282, Dec. 1988.

[11] K. Watabe, S. Sakada, K. Maeno, H. Fukuoka, and T. Ohomri. "Distributed Multiparty Desktop Conferencing: MERMAID". In *Proc. of the Conference on the Computer Support Cooperative Work*, Los Angeles, CA, 1990.

[12] L. G. Palmer and R. S. Palmer. "DECSPIN: A Networked Desktop Videoconferencing Application". *Digital Technical Journal*, Vol. 5(No. 2):pp.65–76., Spring 1993.

[13] M. Stefik, G. Foster, D. Babrow, K. Kahn, S. Lanning, and L. Suchman. "Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving In Meeting". *Communications of the ACM*, Vol.30(No.1):pp.32–47, Jan. 1987.

[14] P. V. Rangan. "Managing Multimedia Collaboration". In *Groupware'92*, pages 418–420, Morgan Kaufmann Publishers, 1992.

[15] T. W. Malone, K. R. Grant, and F. A. Turbak. "The Information Lens: an Intelligent System for Information Sharing in Organization". In *Proc. of the ACM Human Factors in Computing Systems Conference*, pages 1–8, Boston, MA, 1986.

[16] T. W. Malone and K. Lai. "Object Lens: A Spreadsheet for Cooperative Work". In *Proc. of the Conference on Computer Supported Cooperative Work*, Portland, OR, 1988.

[17] J. Brooke. "User Interface for CSCW Systems". *CSCW in Practice: An Introduction and Case Studies*, pages 72–102, Diaper and C. Sanger (Eds.), Germany: Springer-Verlag, 1993.

[18] J. Conklin. "Hypertext: An Introduction and Survey". *Computer, 20(9), IEEE*, pages 17–20; 32–41, September 1987.

[19] R. S. Fish, R. E. Kraut, M.D. Leland, and M. Cohen. "Quilt: A Collaborative Tool for Cooperative Writing". In *Proc. of the Conference on Office Information Systems*, pages 30–37, Palo Alto, CA, 1988.

[20] G. L. Rein and C. A. Ellis. "rIBIS: A Real-Time Group Hypertext System". *Int. Journal of Man-Machine Studies*, pages 339–367, Aug. 1993.

[21] J. M. Haake and B. Wilson. "Supporting Collaborative Writing of Hyperdocuments in SEPIA". In *Proc. of the 1992 CSCW Conference*, pages 138–146, Nov. 1992.

[22] D. R. Jefferson. "Virtual Time". *ACM Trans. Programm. Languages Systems*, Vol.7(No.3):404–425, July 1985.

[23] R. C. Schank. *Conceptual Information Processing*. Amsterdam: North Holland, 1975.

Figure 3: A Snapshot of the Text Editor with a Document Structure and Viewing Windows

Figure 4: A Floor Plan as a Result of Superposition of Two Subplans