Proceedings of the Fourth European Conference on Computer-Supported Cooperative Work,
September 10–14, Stockholm, Sweden
H. Marmolin, Y. Sundblad, and K. Schmidt (Editors)

Contact: Support for Distributed Cooperative Writing

Andrew Kirby
Computing Department, Lancaster University, Lancaster

Tom Rodden
Computing Department, Lancaster University, Lancaster

Abstract: This paper presents a novel system to support the activities of distributed cooperative writing. The system builds upon the results from previous studies of cooperative work, and on a set of short focused studies of cooperative authoring to outline a framework and system to meet the requirements of cooperating authors. The system provides facilities to represent the decomposition of the writing task and assignment of responsibilities. In addition, a series of monitoring facilities is provided which allows authors to coordinate their activities in the construction of documents.

Introduction

Cooperative writing has represented a considerable proportion of research in CSCW and has featured prominently in the literature. In general, work on cooperative authoring has focused on the development of editors intended to support real-time cooperative editing, Grove (Ellis et al., 1991), ShrEdit (Killey 1990) and so on, or on studies of the authoring process (Beck 1993, Beck et al., 1993, Posner 1993, Rimmershaw 1992, Sharples ed 1993). While studies consider the overall strategies and activities involved in the construction and development of documents the current generation of editors focuses on the very detailed set of activities required in the construction of typewritten documents.

As a result of this distinction these two activities often appear to take place in isolation from each other and reported studies of cooperative writing are seldom directly connected to the construction of dedicated cooperative editors. In fact, empirical based arguments are often presented against the narrow scope of dedicated editors :

"support for writing should embrace the entire process from registering the task to supporting a finished manuscript" (Sharples and Pemberton 1990)

Studies of writing do not dismiss or detract from real-time cooperative editors such as Grove (Ellis et al., 1991) or authoring editors like PREP (Neuwirth et al., 1990). Rather they suggest that such editors have a role to play in the wider process of planning and constructing a paper.

The intention of this paper is to examine the development of a cooperative application that directly builds upon the results of existing studies of cooperative work. We take as the starting point in informing our work studies undertaken by (Beck 1993, Posner et al., 1993, Sharples ed 1993). This corpus of empirical material has been complemented by a series of short focused studies of cooperative authoring in an academic environment.

Contact, aims to support the activities of cooperative writing across a community of geographically disparate authors. The focus of support is on the provision of facilities for the decomposition of the writing activity, the assignment of responsibilities and the monitoring of progress. The system makes use of widely available facilities to provide this support to authors within their own environments.

The Needs of Cooperative Writing

In this section we seek to outline the needs of cooperative writing that have driven the construction of Contact. These needs have been gathered from existing studies of cooperative writing across a number of domains. In particular three principal sets of studies have been used to inform our requirements :

- Research Issues in the study of Cooperative Writing (Sharples ed 1993)
- Informed Opportunism (Beck et al., 1993)
- How people write together (Posner et al., 1993)

These studies have been complemented by a short local study of cooperative authoring. The study was conducted at Lancaster University and involved interviewing members of an academic department. These interviewees were drawn from several disciplines and research groups. In general, interviewees were familiar with the other authors and some had often worked with the others on one or more papers.

Interviewees were located in the same building and had regular contact with some of the other interviewees due to the proximity of their offices. In addition, interviewees were also involved in producing collaborative documents with remote participants in the UK and Europe. Interviews were conducted in an informal manner, with an audio tape to record the conversation that took place. The aim of the study was to get a feel for the collaborative processes that the interviewees had taken part in, and not to ascertain any specific data on a particular model or method of work.

Interviewees were asked to focus on the last or most memorable collaborative paper production that they had participated on. The papers discussed included:

- A journal article that was written for a tight deadline, in response to the acceptance of an abstract that was submitted.
- A paper that involved two of the interviewees, and a third collaborator based in Germany.
- A paper for a conference written locally by two of the interviewees who were seldom together in the department.

The mix of papers discussed also covered those working within the same interest or research area, those asked to collaborate due to specialised knowledge from another area, collaborators who had worked together frequently and those collaborators who had not worked together before.

Rather than focus on the particular details of the study we wish to consider the requirements we feel have emerged to drive our software development. In the following sections we consider these requirements in terms of previous studies and of our own short local study.

These recommendations highlight some of the features a system intended to support the dynamic and flexible nature of collaborative writing should have. They are intended to complement the collaborative writing process when supported by electronic tools. In particular, the dynamic, unpredictable and flexible nature of the collaborative writing process is the key factor that will need to be addressed by useful collaborative tools. Consequently any support must allow for considerable flexibility and for considerable control to rest with the user of any supporting systems. This desired flexibility is reflected in many of the existing studies of cooperative work including (Rimmershaw 1992) and (Beck et al., 1993). In the development of our system we wish to focus on two dominant themes which emerge from the investigation of cooperative writing.

Support for coordinating activity

The need for co-authors to coordinate their activities is central to the process of collectively writing documents. For example, (Beck 1993) outlines the need to provide support for coordination which is flexible and open to interpretation by authors. A number of significant points are stressed in the support of coordination:

- The need to integrate with standard platforms given the commitment of users to existing facilities.
- The need to communicate changes across a community of authors.
- The need to provide author information to allow users to infer the impact and consequences of change.

Authors of previous studies also highlight the importance of social control in the development of co-authored documents and the continual change and re-negotiation that this implies. As a consequence, many authors highlight the need for lightweight support.

" We therefore believe that rather than provide generic co-author roles for tools to support, designers might usefully seek to support co-authors exchange of information to help them make their own judgements about appropriate contributions" (Beck et al., 1993, page 246)

The need for an awareness of the activities of others on which to base the development of a document is also reflected in our experiences of studying co-authoring. During the authoring processes studied, there would be periods of time when collaborators could not have regular and easily arranged meetings. This was due to the remote location of the collaborators or the absence of collaborators from their usual place of work. During this time each of the collaborators were expected to carry out work according to their responsibilities in order to be ready to communicate the results of this work to other collaborators. This often led to unease among others who were uncertain how other parts of the document were progressing. As one author expressed "you are reliant upon other people producing their sections."

The need for an awareness of different authors' progress was also reflected in terms of the need to keep the work of the group synchronised. As one author put it " working with busy people .. it is hard not to go badly out of synch .. I hold back, waiting for others to complete their work." Trying to minimise this problem requires some form of management of the different components being worked upon by separate authors. It would be of direct benefit to provide other collaborators with information about the current state of the co-authoring process. This should include:

- When different components may be expected and which have been completed.
- Information to allow a collaborator who has been away from the project to catch up on current progress.

This information promotes a group awareness of the state of the process. Inferences drawn from this information may highlight problems as they occur and support the manual reassigning of responsibilities.

Support for the allocation of responsibility

Our second major theme focuses on the need to allow cooperating authors to divide the task of writing a document into smaller, more manageable chunks. This support needs to recognise the flexible nature evident in a division of labour and that often this division is open to continued re-negotiation. One consequence of this division is that the allocation of work to different users needs to be sufficiently lightweight to allow work to be continually reallocated at low cost.

The need for a flexible approach to the allocation of work requires a careful consideration of the nature of plans and planning. The need to consider the flexibility of plans in cooperative writing is essential. A number of different strategies have been identified in previous studies for partitioning and coordinating writing including what (Sharples ed 1993) calls sequential, reciprocal and parallel. Similar models were suggested by (Thompson 1967) and seen to be employed by groups in studies by (Sharples 1992, Rimmershaw 1992, Kaye 1993 and Posner et al., 1993).

However, the extent to which these models exist in isolation and are followed completely is open to question. As (Beck 1993) states when discussing a detailed study of collaborative writing:

" A view is emerging from the data of the experience of collaborative writing as a process which is dynamically renegotiated " (Beck 1993, page 111)

The need for this flexibility is reflected within the short studies we undertook. However, a clear role existed for the representation of the allocation of sections of the document to users. This representation was either explicit and resulted from a focused effort on planning or implicitly assumed by members of a group. Both the explicit and implicit allocations were reflected by the members of our study. For example, one group stated "we had a meeting to discuss the structure of the document and assign tasks" while another claimed "we each just knew what sections we had to write."

In all of the recorded collaborative writing experiences some form of outline plan was used to divide the work among the collaborators, to provide a set of goals to be achieved by individuals or to describe the steps that must be taken to complete the writing process. These were interpreted quite loosely and often the status of the work against the plan was inferred from considerable knowledge of the individuals involved.

The use of explicit plans detailing which responsibilities each individual had were most commonly used when the group was remotely located. In developing support for such planned activity we need to make available responsibilities throughout the writing process. However, the enforcement or imposition of a plan is inherently social, and support must take account of the dynamic nature of the writing process and be flexible enough to allow frequent changes in line with the changes of group membership and individuals' capabilities to achieve their assigned tasks.

The Contact Approach

The core of our approach is the development of mechanisms that make visible the current allocation of responsibility and the activities that need to be coordinated across a group. Our starting point is the electronic context within which the actions involved in cooperative writing take place. Rather than consider the representation of these activities abstractly we wish to focus on the ways in which the endeavour of cooperative writing is manifest electronically among the community of users. The cooperative writing of documents involves a number of heterogeneous tools which are used in tandem to manipulate stored artifacts. When the group is distributed these often exist across a wide variety of machines and much of the effort of cooperative writing involves understanding and managing the ways in which a document is spread across these different electronic stores. For example, to write a bibliography for a paper an author will interact with several tools to generate the text of the bibliography. Constructing the bibliography may also require knowledge of which authors are responsible for constituent parts of a document and where these are located. Often remote information stores cannot be directly accessed and users need some form of version system to control the exchange of replicated document copies. Active contexts for a project are formed by aggregating all the resources of local contexts for each user. Users may form the local active context by associating local resources with the writing project. Alternatively these associations maybe given to them from a centralised server.

In line with previous approaches we seek to develop techniques which allow actions involved in cooperative activity to be monitored or recorded (Beck 1993, Kreifelts et al., 1991, Sarin et al., 1991, Trigg et al., 1986). The increased access to component parts of documents will reduce the need for version control. Members of the group may use the information concerning the interactions with stored information to make inferences about the overall state of the activity. From the initiation of the activity, as soon as an interaction occurs with any of the resources in the activity context the state of the activity could be inferred to be started. A period of sustained interactions with the resources within the context could indicate that the action was being performed or executed. If after such a period there was then a period of inactivity with any of the resources, then perhaps the action has been completed or halted, unable to be completed. It is clear that the captured interactions do indicate information about how the activity is being performed or conducted, and that some inferences can be drawn about the state of an individual activity based upon these captured interactions.

The actions on shared objects also serve as an overall indicator of the state of the document development process across its lifetime. From its planned inception until the first recorded interaction with a component, the development of a document remains initiated, but not started. When the first interaction occurs the process starts and while interactions occur then the process can be inferred to be under execution, until all interactions cease, when the action may be deemed complete or finished. This is obviously a simplified view of the nature of interactions over time, and the process of starting and completing can be considerably more complex. The point to make is that many of these subtleties are subject to the particular circumstances and can be inferred by the community involved in the cooperative development of the document.

In addition to publicising the interactions with shared resources, the available attributes of the resources can be used to infer information about the state of the shared endeavour. In particular we wish to initially focus on the existing attributes of electronic files such as size, owner, and write time. Each of these attributes offers the possibility of inferring different information about the state of the cooperative activity For example, a file's last write time indicates the last time any modifications were made to the file. This may allow a user to determine within the context of cooperative development if a component has altered in some significant way.

Each of these attributes can be used only within a given context and users may wish to view particular attributes as significant and may wish to be informed when these attributes alter. For example, within our small scale field studies one user commented "I know that he takes care and time to construct what he wishes to say but when he does he writes incredibly quickly and it seldom needs much editing". This use could make considerable use of the size attribute of a file to indicate when his colleague had begun writing his component.

Similarly within a given context users may introduce measures to estimate the overall level of completion of an activity and these measures can be checked against the current document. For example, we found that members of a writing group often used word count as a rough guide to the level of completeness of component parts. A document meeting agreed measures may indicate that the activity is nearing completion. The use of these measures can be used in conjunction with the level of interaction to add weight to any inference.

Our principle aim is to provide the mechanisms to enable the capture of this interaction and the value of attributes with shared resources. We achieve this by making the resources used in the construction of a shared document active. To encourage heterogeneity we have chosen to adopt a low tech approach and have restricted the scope of the resources. In particular, we endeavour to make only the electronic files representing the data resources, and the results of commands and tools used to manipulate these publicly available. In adopting our minimalist approach we do not consider the internal effects of tools on the structure and content of documents since this will require the general use of a common editor. A number of authors highlight the many benefits of a single system approach and highlight the advantages of technologies such as Hypertext to represent the structure of documents. For example, SEPIA (Haake et al., 1992) offers sophisticated internal support for representing the cooperative structure of documents.

Our philosophy is to provide support for the heterogeneous collection of editors which our user community has invested in. In this respect our aims are similar to those of the MESSIE developers (Sasse et al., 1993). However, we focus on more readily available common access to shared components making up the document under construction. The core of our approach rest with three inter-related components:

- A decomposition tool wich assigns portions of a writing activity to shared information resources.
- A framework which monitors interaction with shared objects and propagates these across the user community.
- An association facility which links the action of specific tools on objects with events significant to the user community.

These different components work in tandem to provide lightweight monitoring support across a community of users.

The Developed System

The basis of our approach is the development of mechanisms which promote a shared awareness of a common electronic context within which the work is taking place. A set of assumptions about the supporting environment which constitutes the electronic context are critical to the systems development. These assumptions include:

- Each of the users involved in the cooperative endeavour has regular access to a computer environment for conduction of the work.
- The computer environments used will normally be distributed across a range of diverse machine types.
- Users will have access to at least EMail facilities as a means of remote electronic communication.
- Any developed system needs to be used alongside existing systems or provide mechanisms to allow integration with existing systems.

Support is provided through a toolkit which supports the propagation of effects on shared objects. To allow the toolkit to operate across a diverse range of platforms a set of shared resources are made commonly available. These include the files and tools that are the subject of interest, and the object data for the toolkit. The toolkit data is accessed through an object interface that allows for extendibility and the future inclusion of a database to store toolkit data. The interaction monitors within the environment are the principle mechanisms by which interaction with the resources is captured. These communicate the captured data to the toolkit managers, locally and when at a remote site through any available TCP/IP distributed communications platforms.

The toolkit managers are process objects responsible for the maintenance of data in the system, the control of active contexts. They also provide the interface for applications wishing to use the Contact platform. The current toolkit contains the following components:

- *A Project Editor* responsible for the initialisation of the project, maintenance of the project group membership and members' data. It provides an interface allowing other applications to initiate a new project.
- *A Component Editor* responsible for the list of resources making up the context. Like the project editor it offers an interface to allow applications to edit project components and their active contexts.
- *A Report Object* responsible for communicating the interactions monitored by watching agents to a server message object. The report object ensures that routing occurs to the correct message object, and also provides a damping effect for multiple interactions with resources that occur over a very short period.
- *A Message Agent* responsible for communication and data passing between the remotely located elements of the system. The message agent is independent of any supporting distributed platform. Communication is via a dedicated communication object that uses its own protocol to communicate with the message agent on top of the underlying platform. This allows a communications object to be written for the actual platform that is in use or available between the remotely located areas of the collaboration.

A writing project in the system is represented by a decomposition list where the activity of writing is divided into different components. The system allows particular users to become responsible for components within the project. This form of rough outlining and allocation of responsibilities figured significantly in the development of documents we observed and it is widely reported in the writing literature. This approach has also been used to present activity based systems, for example the Task Manger (Kreilfelts at el., 1993). In addition to assigning responsibilities we also allow local electronic resources to be associated with parts of the project. Normally, this takes the form of a file containing a portion of the document. In addition, we allow users to make public commands invoked on these objects. In the case of UNIX platforms this includes commands to spell and count words in a file.

The active context within the system is supported and maintained by three distinct system elements, the "local objects", "server objects" and a set of user interfaces. Local objects are located on work site machines, and their primary purpose is to maintain the associations that describe members' working contexts. For each component of the project the local user has responsibility, two lists are maintained, one listing the associated files and the other the associated commands.

Server objects focus on distributing details of the project and recording the captured interactions from local objects. They also maintain data describing the collection of components in each project, and the membership of the writing project. It is here that the associations between members and components are maintained. At present two forms of user interface to the system have been developed, an X Windows interface, and a World Wide Web interface.

Server Objects and Data

Any project in the system consists of the collection of components created for that project; the membership of the project and some optional data, such as a deadline for completion of the project. This information is held in a project record, and the project editor is responsible for any amendments to existing projects, or for creating new projects. It also contains a message object which is responsible for distributing any amendments or new component records to local message objects.

The Context server objects are responsible for initiating component parts of the project and recording the reported interactions from component's active contexts. Figure 1 shows the objects located at an Active Context Server. Only the Message Server object is persistent, the others are invoked as needed.

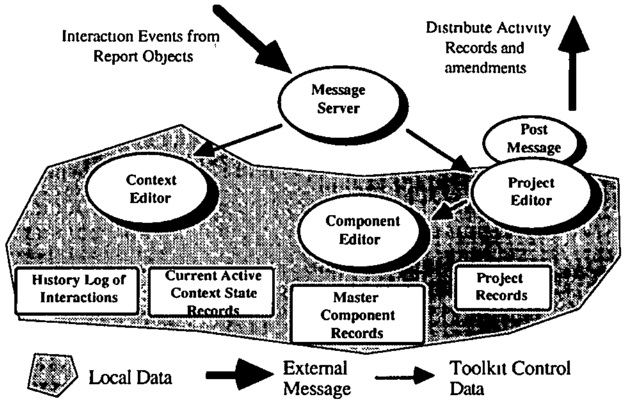


Figure 1 The Active Context Server Objects.

The Message Server is responsible for accepting all interaction events, decoding the source message and passing the data to the Context Editor. It also decodes information about changes in the component records from remote locations and passes these to the project editor.

The Context Editor is responsible for maintaining the two Active context records, the History Log and the Current Active Context Record. The History Log is an archive of event interactions that have taken place. The current active context record provides an overview of the resources within a particular component's active context. It displays the resources associated with a component along with additional information such as the last time an interaction took place. The record also maintains state to indicate which resources in an active context have been used since the last time the record was viewed. The context editor also provides an interface allowing the records to be viewed by other applications.

The use of message objects both for sending and receiving information between the local and server domains is motivated by the need to support the different platforms and operating systems members may use. The message object is independant of any supporting distributed platform. This is achieved by making use of individual specific communications objects, each communications object is written for a specific distributed communications platform or media.

Local Objects and Data

Four principal objects are used at local sites to support the system as shown in Figure 2. The Local Message Receiver and Watch Control Objects are persistent processes, while the Report Object and Component Editor are invoked as required. Local data consists of a record for each component that the member is responsible for, and two local lists specifying the files and commands that are part of the working context.

The Local Message Receiver is responsible for accepting and decoding messages from the project server. These messages are usually amendments to the component information and any required updates are carried out. In addition the component editor allows access to the local component information and the list of resources making up the active context. The Watch Control Object is responsible for creating and destroying watch objects. Watch Objects are created to monitor individual resources, and report when interactions occur or the resource is used. Our initial implementation has focused on UNIX and two types of watch object exist at the moment, one to detect alterations to files, and the other to register commands invoked upon files.

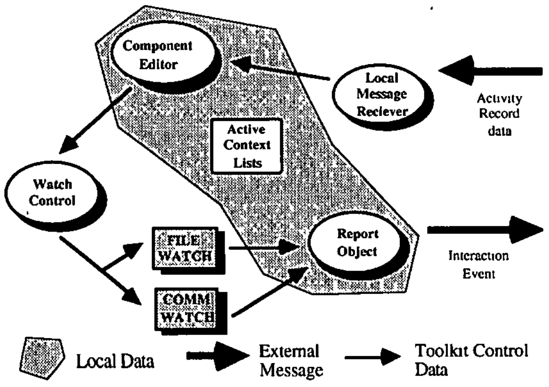


Figure 2 Showing the Local Objects and the interactions between Local Objects

The Report Object is responsible for communicating the interactions monitored by the Watch Objects to the project server. The Report Object is used to communicate with the project server rather than allow direct communication from Watch Objects. As a result, only one Watch Object is required per resource, as multiple associations are handled by the Report Object. It also provides a damping effect for multiple interactions with resources that occur over very short time periods, where one interaction notification is sent rather than several.

One of the key design goals for Local Objects was to allow for multiple platforms or operating systems to be supported. Consequently, the Report Object, Component Editor and Watch Control Objects make no direct calls to the underlying operating system. Any system dependant code has been structured within a set of library functions called *sysutils*. This allows us to write a set of sysutils for each platform that the system may be placed on, and limit changes to the above objects. The File Watch and command Watch Objects are platform specific as they rely upon interrogating detailed operating system resources. To aid portability Watch Objects can be developed for different platforms. We have developed UNIX Watch Objects and are currently constructing watch objects for Macintosh platforms.

The System in Use

To demonstrate how the Contact system can be used we will describe a simple scenario based upon the generation of this paper. We will present a series of snapshots of web pages presented by the Contact system during the lifetime of the project. The web interface allows us to promote ubiquitous and heterogeneous access to the system.

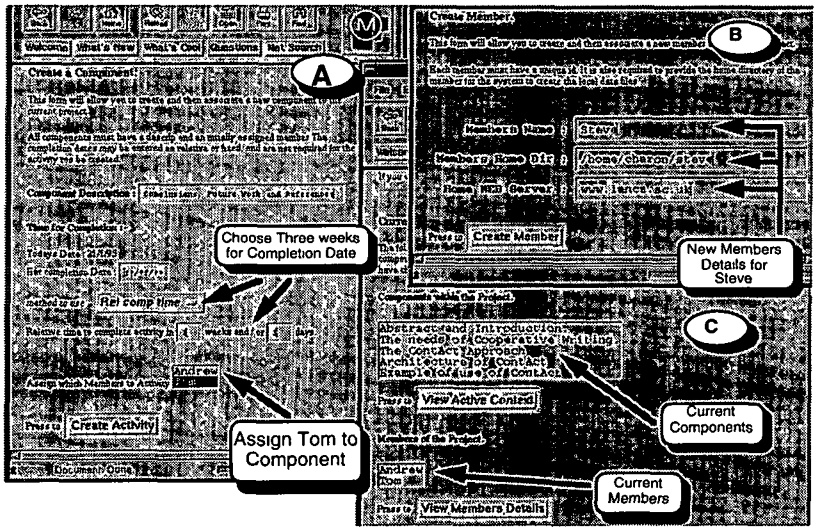
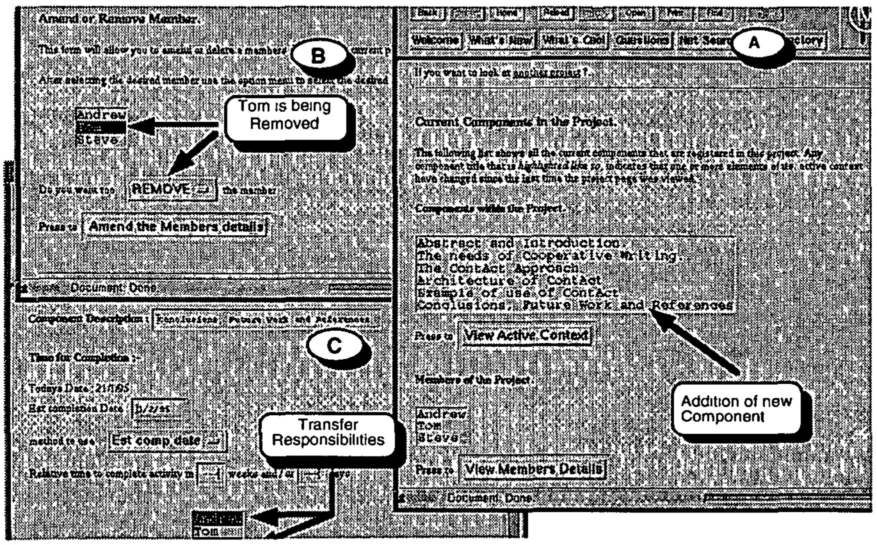


Figure 3 Snapshot 1, the addition of a new member and component to the project.

The Project involves the writing of a conference paper describing the system and its design goals. The initial task is therefore to initiate a project, called *Contact: Support for Distributed Cooperative Writing.* There are two members in the group, Andrew and Tom. After requesting Contact to initiate a new project, it is possible to start the decomposition of the writing process into components. The details of the two initial members, Andrew and Tom are entered using form interface (**B**) shown in Figure 3. The initial data allows Contact to create Local Object data so that local active context lists can be created. The initial components of the project represent the various sections of the document to be written. At the time of snapshot 1 five components have been created and assigned between Andrew and Tom (shown in form (**C**)). At this point it is decided that another person is required to help with the project, as Tom is often difficult to find. Form (**B**) shows the creation of a new member associated with the project called Steve. In form (**A**) Tom also creates another component, *Conclusions, Related Work and References,* with an initial completion target date of 3 weeks.

As each component is created, Contact will pass a component record to the Local Objects of each responsible member. At each members local site, they may assign whichever files and commands they will use for each component. Tom assigns the file ContactDraft.1 and his choice of editor Xedit to the component *Conclusions, Related Work and References.* Similarly each of the other users assign whichever resources they will be using.


Amend or Remove Member.

This form will allow you to amend or delete a members current p

After selecting the desired member use the option menu to select the desired

Andrew
Steve

Do you want too REMOVE the member

Press to Amend the Members details

Document: Done

Component Description : Conclusions, Future Work and References

Time for Completion :-

Todays Date 21/1/95

Est completion Date 11/2/95

method to use Est comp date

Relative time to complete activity in [illegible] weeks and/or [illegible] days

Tom

If you want to look at another project ?

Current Components in the Project.

The following list shows all the current components that are registered in this project. Any component title that is *highlighted like so,* indicates that one or more elements of its active context have changed since the last time the project page was viewed.

Components within the Project.

Abstract and Introduction.
The needs of Cooperative Writing.
The ContAct Approach
Architecture of ContAct
Example of use of ContAct
Conclusions, Future Work and References

Press to View Active Context

Members of the Project.

Andrew
Tom
Steve

Press to View Members Details

Document: Done

Figure 4 Snapshot 2 showing the transfer of responsibilities between members.

The project then proceeds until Tom is called away. At this point because the component *Conclusions, Related Work and References,* has not been completed there is a need to reassign responsibility to others in the group. In snapshot 2, Fig 4. Tom decides to remove hımself from the project, and can be seen doing this in form **(B).** The current components within the project are displayed in form **(A)**, and the reassignment of responsibilıties is conducted in form **(C).** The re-assıgnment can be performed by any of the members either accepting responsibility, or by Tom giving responsibility to another member. Sınce the deadline for the completion of the project is nearing, both Andrew and Steve are given responsibility. Negotiation of responsibilities is assumed to take place socially and is not driven by the system.

It is decided that Andrew should continue working on the file ContactDraft.1 and a new copy, ContactDraft.2 will be used by Steve. Andrew associates the spell command with the active context as he nears completion of ContactDraft.1 and he begins to proof check the document. Steve ıs waiting on Andrew to complete his work before integrating the two drafts with the references file he has written. He knows from previous experiences of working with Andrew that once the spell command has been used successfully on ContactDraft.1 that Andrew will probably have completed his work. Steve consults the current active context state for component *Conclusions, Related Work and References,* shown in Fig 5, until he notices that the spell command has been used, but no changes were made to the file ContactDraft.1. Steve infers that the draft is complete, confirms this with Andrew and begins to integrate the drafts to create a final version of the paper.

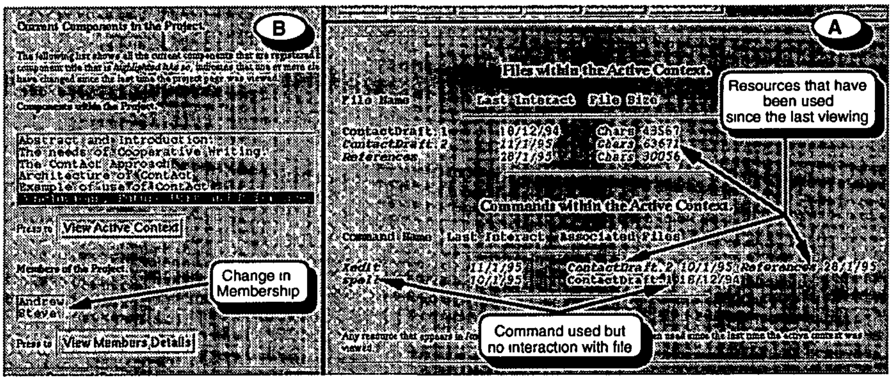

B
A
Current Components in the Project
Components within the Project
View Active Context
Members of the Project
Andrew
Steve
View Members Details
Change ın Membership
Files within the Active Context
File Name
Last Interact
File Size
Resources that have been used since the last viewing
Commands within the Active Context
Command Name
Associated Files
Command used but no ınteraction with file

Figure 5 Snapshot 3 showing the use of the resources withın the component *Conclusions, Related Work and References.*

Conclusions and Future Work

We have identified and attempted to address some of the issues raised in studies of cooperative authoring. In particular we have focused on providing a lightweight framework within which members of a collaborative authoring project can decompose the writing task, and freely assign responsibilities for the tasks in a flexible and unconstrained manner. We also provide users with a facility to construct an active context for the project. This context is composed of the electronic resources used by members of the group in the process of completing their work. By interrogating this active context we allow users to draw inferences on the state of any component activity within the project. This offers a greater degree of "author information" and increases the possibility for coordination between the members.

A study of the implementation and use of our system will provide us with more empirical data on the nature and characteristics of cooperative writing. However, at this point we already believe that the use of Contact may be beneficial in other domains. In highly distributed project environments, where many sub-components are in use at any time, Contact will allow a supervisor to monitor which sub-components are used or accessed. In any highly versioned situation, Contact may also make users aware if old or abandoned versions are still being used. One of the main design features of the Contact toolkit was to allow its use with third party software. We are particularly interested in its use with Workflow systems and shared object servers.

At present the Contact system is fairly passive, its use of the web relies on users interrogating active contexts and drawing conclusions based upon the latest set of interactions. We wish to make the toolkit more proactive, so that a user can be informed when a specified state has arisen. For example the user may request to be informed when a component's file has reached a set word length, or a particular editor has been used in conjunction with two files. To enable this we are developing trigger mechanisms for the Contact server that will notify users when particular interaction patterns have occurred. As part of this endeavour we are exploring how to modify the Web server to make it more active and to allow greater integration and functionality with Contact.

References

Beck, E E (1993) A survey of Experiences of Collaborative Writing In Computer supported Collaborative Writing, ed. M Sharples, London: Springer-Verlag.

Beck, E.E and M.E. Bellotti. (1993) Informed Opportunism as Strategy· Supporting Coordination in Distributed Collaborative Writing. In ECSCW 93, Milan Italy, pp 233-247.

Ellis, C.A. , S.J. Gibbs, and G.L. Rein (1991) Groupware. Some Issues and Experiences. Communications of the ACM 34 (1): pp 38-58

Haake, J.M. and B. Wilson. (1992) Supporting Collaborative Writing of Hyperdocuments in SEPIA. In CSCW 92, Toronto Canada, pp 138-147.

Kaye, A.R. (1993) Computer Networking for Development of Distance Education courses. In *Computer Supported Collaborative Writing.*, ed. M Sharples, London: Springer-Verlag

Killey, L. (1990) ShrEdit 1.0. A shared editor for macintosh. Cognitive science and Machine Intelligence Laboratory, University of Michigan, 1990

Kriefelts, T , E Hinrichs and G. Woetzel. (1993) Sharing To-Do lists with a Distributed Task Manager In ECSCW 93, Milan Italy, pp 31-46.

Kreifelts, T., U. Pankoke-Babatz, and F. Victor. (1991) A Model for the Coordination of Cooperative Activities In Proceedings of the International Workshop on CSCW, Berlin, pp 85-100

Leland, M.D P. , R S. Fish, and R E Kraut. (1988) Collaborative Document Production using Quilt. In CSCW 1988, Portland Oregeon, pp 206-215.

Miles, V C , J C McCarthy, A.J Dix, M.D Harrison, and A F. Monk. (1993) Reviewing Designs for a Synchronous-Asynchronous Group Editing Environment. In Computer supported Collaborative Writing, ed M Sharples, London Springer-Verlag

Neuwirth, C.M. , D.S. Kaufer, R Chandok, and J H Morris. (1990) Issues in the Design of computer support for co-authoring and commenting. In CSCW 90, pp 183-195

Posner, I.R. and R.M. Baecker (1993) How people Write Together In Readings in Groupware and Computer Supported Cooperative Work., ed R.M. Baecker, pp 239-250. Morgan Kaufmann.

Rimmershaw, R. (1992) Technologies of Collaboration. In Computers and writing. Issues and Implementations, ed. M. Sharples, Dordrecht: Kluwer

Sarin, S K , Abbott, K.R., and McCarthy, D R (1991) A process model and system for supporting collaborative work In COOCS 91, pp 213-224

Sasse, M.A. and M.J. Handley. (1993) Support for Collabortive Authoring via Email The MESSIE Environment. In *ECSCW 95, Milan Italy*, pp 249-264

Sharples, M (1992) Adding a little structure to collaborative writing In CSCW in practice: An introduction and Case Studies, ed D Diaper and C Sanger, London· Springer-Verlag.

Sharples, M. ed (1993) Computer supported Collaborative Writing London: Springer-Verlag

Sharples and Pemberton (1990). Starting from the writer. guidelines for the design of user centred document processors Computer Assisted Language Learning 2, pp 37-57.

Thompson, J.D (1967) *Organisations in Action* New York. McGraw-Hill

Trigg, R H , L A Suchman, and F.G. Halasz. (1986) Supporting collaboration in NoteCards In CSCW 86, Austin Texas, pp 153-163.