

THE UNIVERSITY OF CALGARY

# **GroupWriter**

## **A Word Processor for Collaborative Document Preparation**

BY

NICHOLAS MALCOLM

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE  
DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

APRIL, 1991

© NICHOLAS MALCOLM 1991



National Library  
of Canada

Bibliothèque nationale  
du Canada

Canadian Theses Service    Service des thèses canadiennes

Ottawa, Canada  
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

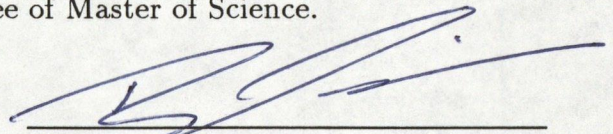
ISBN 0-315-66966-7

Canada

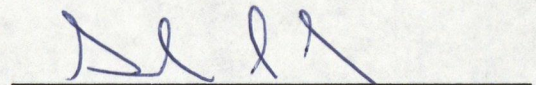


**The University of Calgary**  
**Faculty of Graduate Studies**

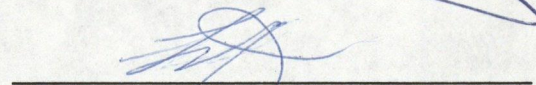
The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "*GroupWriter: A Word Processor for Collaborative Document Preparation*" submitted by Nicholas Malcolm in partial fulfillment of the requirements for the degree of Master of Science.



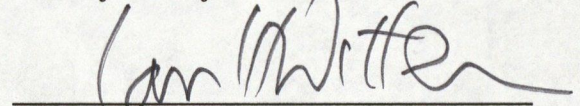
Supervisor, Dr. Brian Gaines  
Department of Computer Science



Dr. Saul Greenberg  
Department of Computer Science



Dr. Larry Katz  
Faculty of Physical Education



Dr. Ian Witten  
Department of Computer Science

Date 1991-04-19

## Abstract

Group-writing, in which a document is produced jointly by a team of writers, occurs widely in both science and industry. Group-writing is both time and labor intensive, and is not specifically supported by current generation word processors. Consequently there is much interest in developing computer-based tools to support group-writing. This thesis surveys some of the issues behind group-writing—its occurrence in the scientific community, and cognitive aspects of writing—in order to derive some requirements which should be satisfied by such a system. A prototype group-writing system is then presented, which can serve as a basis for further experiments on how best to provide computer support for group-writing.



## Acknowledgements

First of all my thanks and appreciation to my supervisor Brian Gaines. His guidance, helpful suggestions, and wide-ranging knowledge not only made this thesis possible but also gave me a much better understanding of the issues involved.

Second, I extend my warmest thanks to all those who offered suggestions or encouragement regarding the work presented in this thesis: Dan Freedman, Saul Greenberg, Larry Katz, Tamara Lee, Veronica Maltin, Ben Schneiderman, Jinqiu Shao, Maurice Sharp, Mildred Shaw, and Camille Sinanan.

Finally, my sincerest thanks to my friends and family who have not only borne with me over the past months, but even offered positive support: Rose Caday and Haihuai Chen in Canada; Wei Zhao in the U.S.; Lisa Nayda, Andrew Thompson, Pia Williams, Li-Jun Yao, and, most importantly, my family, home in Australia.

# Contents

Approval	ii
Abstract	iii
Acknowledgements	iv
List of Tables	vii
List of Figures	viii
<b>1 Introduction</b>	<b>1</b>
<b>2 Background Work</b>	<b>4</b>
2.1 Professional Collaboration . . . . .	4
2.1.1 Collaboration in Science . . . . .	4
2.1.2 Collaboration in Business . . . . .	11
2.2 The Writing Process . . . . .	11
2.3 General Requirements for a Group-Writing System . . . . .	20
2.4 Specific Requirements for a Group-Writing System . . . . .	21
<b>3 Related Systems</b>	<b>24</b>
3.1 Computer Support of the Writing Process . . . . .	24
3.2 Group-Writing Tools . . . . .	27
3.2.1 Linear Systems . . . . .	27
3.2.2 Hypertext-based Systems . . . . .	32
3.2.3 A Comparison of Group-Writing Systems . . . . .	38
<b>4 Document Storage</b>	<b>40</b>
<b>5 Overview of GroupWriter</b>	<b>43</b>
5.1 GroupWriter in Brief . . . . .	43
5.2 GroupWriter in Detail . . . . .	44
5.2.1 Basic Editing Operations . . . . .	44
5.2.2 Comparing Different Document Versions . . . . .	46
5.2.3 Annotation Facilities . . . . .	52
5.2.4 The Graphical Browser . . . . .	52
5.2.5 Import/Export of Document Versions . . . . .	56

<b>6</b>	<b>Implementation of GroupWriter</b>	<b>58</b>
6.1	Overall Architecture . . . . .	58
6.2	User Interface Design . . . . .	58
6.3	Detailed Architectural Description . . . . .	61
6.3.1	Storage Management . . . . .	61
6.3.2	Paragraph Management . . . . .	65
<b>7</b>	<b>Evaluation of GroupWriter</b>	<b>69</b>
7.1	Comparison with RCS . . . . .	69
7.2	Preliminary User Trials . . . . .	71
7.2.1	Questionnaire Results . . . . .	72
7.2.2	Trial Use of GroupWriter . . . . .	77
<b>8</b>	<b>Conclusions and Future Work</b>	<b>79</b>
8.1	Future Work . . . . .	79
8.1.1	Apple System 7.0 . . . . .	79
8.1.2	Document Storage . . . . .	80
8.1.3	Improvement of Existing Features . . . . .	81
8.1.4	Additional Features . . . . .	85
8.1.5	Document Version Comparison . . . . .	86
8.1.6	User Trials . . . . .	87
8.2	Conclusions . . . . .	87
	<b>Bibliography</b>	<b>89</b>
<b>A</b>	<b>User Survey Questionnaire</b>	<b>98</b>

## List of Tables

2.1	Collaboration in Different Disciplines . . . . .	7
3.1	Group Writing Systems . . . . .	28
3.2	A Comparison of Group-Writing Systems . . . . .	39
7.1	Comparison of GroupWriter and RCS: Case I. All measurements are given in bytes. . . . .	70
7.2	Comparison of GroupWriter and RCS: Case II. All measurements are given in bytes. . . . .	71
7.3	User Profiles . . . . .	73
7.4	Previous Experience . . . . .	73
7.5	Importance of Various Factors to Group-Writing (on a scale of 1 to 10)	74
7.6	General Responses to the Questionnaire . . . . .	75
8.1	A Comparison of Group-Writing Systems . . . . .	88



## List of Figures

2.1	The Flower and Hayes Model of the Writing Process . . . . .	12
2.2	The Horizontal Division Model of Group-Writing . . . . .	17
2.3	The Sequential Model of Group-Writing . . . . .	18
2.4	The Stratification Model of Group-Writing . . . . .	19
3.1	Document Preparation with a Pure Formatter . . . . .	25
3.2	An RCS Version Tree . . . . .	32
4.1	Free Branching Objects in Cosmos . . . . .	42
5.1	Opening a Document Version . . . . .	45
5.2	Branches in the Version History . . . . .	46
5.3	Paragraph Bracketing . . . . .	48
5.4	The Popup Menu and Alternative Paragraphs . . . . .	49
5.5	Viewing the Difference between two Paragraphs . . . . .	51
5.6	Annotating a Paragraph . . . . .	53
5.7	The Popup Menu with an Annotation . . . . .	54
5.8	The Document Version Browser . . . . .	55
5.9	Exporting a Document Version . . . . .	57
5.10	The File Format Dialog . . . . .	57
6.1	The Architecture of GroupWriter . . . . .	59
6.2	Storing a Document Version . . . . .	63
8.1	The Help Manager . . . . .	81

# Chapter 1

## Introduction

Group-writing—the joint production of a document by a team of writers—occurs widely both in science and in industry. Group-writing is both time and labor intensive, and is not specifically supported by current generation word processors. Consequently, there is much interest in developing computer-based tools to support group-writing. This thesis examines the requirements a group-writing system should satisfy, and describes the implementation of *Group Writer*, a prototype group-writing system.

There have been numerous studies examining collaborative trends within the scientific community. These studies usually use the number of coauthors of a paper to measure collaboration, and the results show that the trend is towards more collaboration; more papers are being collaboratively written. In the business community too, collaborative writing is widespread, whether it be ghostwriting or writing annual reports.

As will be shown in the next section, writing itself is a complex and demanding activity. Further, the act of writing is influenced by the medium being used; a word processor will promote different writing styles than pen and paper. Additionally, many members of group-writing teams have already had experience with traditional word processors such as Microsoft Word. The difficulty of writing, together with user familiarity with existing word processors, suggests that a group-writing program should be both simple to use and similar to existing word processors. Ideally, writers

should be distracted as little as possible from the process of writing, and should be able to transfer already learned skills instead of learning new ones.

This thesis describes the use and implementation of GroupWriter, a new word processor for group-writing. GroupWriter offers a number of facilities:

- Multiple parallel versions of a document may exist at the same time, and several users may work simultaneously on the same version of a document.
- A complete record of all versions of a document is maintained in an archive.
- A graphical browser is provided to examine the relationships between document versions.
- Paragraphs may be annotated.
- Alternative versions of paragraphs in the document version open for editing may be found in other document versions and made available for editing.

GroupWriter is currently implemented in prototype form for the Apple Macintosh series of computers. Because providing computer support for group-writing is a relatively new area, it is important that an iterative approach be adopted in the design of such a system. Implementation should repeatedly alternate with user evaluation [GBL91], resulting in the system being iteratively improved. Accordingly, the results of some preliminary user trials are presented in this thesis, and a discussion of the resultant benefits.

The aims of this thesis are threefold:

1. To formulate a reasonable set of requirements to be met by a group-writing system if it is to be widely used.

2. To implement a prototype group-writing system that can be used as a basis for further experimentation.
3. To perform some preliminary user trials with the group-writing system.

The remainder of this thesis is organized as follows. Chapter 2 contains a review of some of the areas related to group-writing: collaborative trends in science and in industry; a brief examination of cognitive aspects of writing; and the impact of word processing on the writing process. Chapter 3 reviews other systems that have been developed to support group-writing, and presents a classification scheme in which to regard them. Chapter 4 discusses the storage scheme used in GroupWriter. Chapter 5 contains an overview of the features offered by, and the use of, GroupWriter. Chapter 6 describes the implementation of GroupWriter. Chapter 7 contains a comparison of the storage efficiency of GroupWriter and RCS, and a description of some user comments regarding GroupWriter. Chapter 8 contains the conclusion and some suggestions for future work.

## Chapter 2

### Background Work

This chapter explores some of the background issues that provide motivation for constructing a group-writing system and that guide its design. First, collaborative trends both in the business and in particular the scientific communities are examined. The incidence of group-writing within these communities is seen to be increasing. Second, the writing process itself is examined. Cognitive models of writing are described, along with a review of the effect of word processing on the writer. All these factors are then combined to derive some requirements for a group-writing system.

#### 2.1 Professional Collaboration

The following sections contain a discussion of collaborative trends in science and in industry.

##### 2.1.1 Collaboration in Science

Science is a collaborative process, with researchers building on results obtained by their forerunners. In a more concrete sense, however, scientific collaboration occurs when two or more researchers work together on a project, contributing both intellectual and physical resources and effort [Sub83a]. These collaborative efforts often lead to the collaborative production of a document for publication. Several types of

scientific collaboration can be identified [Sub83a]:

- *Teacher-pupil collaboration.*
- *Collaboration among colleagues.*
- *Supervisor-assistant collaboration.*
- *Researcher-consultant collaboration.*
- *Collaboration between organizations.*
- *International collaboration.*

In addition, the level of collaboration may vary from general advice and opinions to active and sustained participation in a research project. The degree of collaboration is generally higher in scientific and in technical fields, and lower in the humanities [Sub83b].

### **Trends in Research Collaboration**

Collection of data concerning research collaboration is difficult [Sub83a], because forming a qualitative assessment of the contributions of each collaborator, or a quantitative assessment of advice, of ideas, and of criticism, is problematical. Consequently, in examining the extent of collaboration in science, many researchers use the number of coauthors of a research paper as a convenient measure of research collaboration. Measuring collaboration in this way has several advantages: it is invariant; it is easily and inexpensively ascertainable; it is quantifiable; it is non-reactive (the process of estimating the degree of collaboration does not affect the process of collaboration itself) [Sub83a].



Nonetheless, this bibliometric method does have some disadvantages, in that it implicitly makes the following assumptions [Sub83a]:

1. The number of papers produced by a research group is proportional to its research activity.
2. The relative frequency of coauthorship within such groups is proportional to the degree of collaboration within the group.
3. The relative frequency of production of research papers with differing amounts of multiple authorship is proportional to the relative frequency of acceptance of papers by groups of each size.

These assumptions are difficult to verify. Regarding the third assumption, Gordon [Gor80] (cited in [Sub83a]) found that in astronomy there is a significant relationship between the number of authors per paper and the rate of acceptance for publication. Papers with a higher number of coauthors have a higher rate of acceptance for publication. Gordon conjectured that the same is true for disciplines where highly complex, large scale equipment is required for experimentation and observation, as in high energy physics.

Bibliometric studies of the amount of collaboration in scientific research show that the trend is towards increasing amounts of collaboration. Over [Ove82] has found that the mean number of authors for papers published in journals of the American Psychological Association rose from 1.47 in 1949, 1.72 in 1959, 1.88 in 1969, to 2.19 in 1979. The percentage of single-author papers declined in the same period.

The degree of collaboration varies from one field to another. Table 2.1 (modified from [Sub83b]) shows the proportion of multiple-author papers and the average

Discipline	Total number of papers in the sample	Proportion of multiple- author papers	Average number of authors (for all papers)
Biochemistry	2880	91%	2.7
Chem. Engineering	530	86%	2.1
Computer Science	6148	43%	1.7
Mathematics	Unknown	21%	Unknown

Table 2.1: Collaboration in Different Disciplines

number of authors per paper for biochemistry, chemical engineering, computer science, and mathematics. Collaboration in mathematics is low; over 78 percent of the papers have only one author. In contrast, biochemistry is a highly collaborative discipline with less than 10 percent of the papers having only one author. Furthermore, Pao [Pao82] has examined the degree of collaboration in a humanistic subject—musicology—and found that although only 15 percent of articles are multiple author, the most collaborative researchers are also the most productive (the inverse relationship does not hold).

Why is the amount of research collaboration increasing? Many explanations have been forwarded [Gor80], including the rationalization of scientific manpower, the advent of “Big Science”, patterns of research funding, the increasing specialization of science, and the professionalization of science. Some of these explanations will be considered briefly here, though the true answer is probably more complex, involving a number of factors.

Heffner [Sub83a] has studied the relationship between the level of collaboration and the level of financial support in the fields of political science, psychology, biological science, and chemistry. In all these fields, Heffner found that an increase in

funding is associated with an increase in the number of authors per paper, particularly in biological science and chemistry.

Another explanation for the increasing amount of scientific collaboration is the increasing specialization of science [dBR78]—as science becomes increasingly specialized, no individual researcher can completely master separate scientific disciplines. Hence teams of researchers are required in many projects. This explanation does not account for the widespread occurrence of collaboration before science became specialized. Nor does it explain why some disciplines, such as biochemistry, are highly collaborative, and others, such as mathematics, are not. Mathematics is not unspecialized when compared to biochemistry.

Beaver and Rosen [dBR78, dBR79a, dBR79b] offer a more general explanation, proposing that collaboration is a response to the increasing support of the scientific community by outside society—a response to the professionalization of science. Collaboration provides a means to professional advancement, to increased knowledge, and to better research facilities. In those fields where this is more true, the degree of collaboration will be higher.

They show that scientific collaboration was low in Europe during the 17th and 18th centuries, at which time there were few professional scientists. They further contrast the amount of collaboration in the French scientific community during the period 1790 to 1830 with that in the British and the German scientific communities. The French scientific community received substantial state support during the Revolutionary and the Napoleonic eras (though this process was well under way during the final years of the monarchy [Sch89]), whereas the British and the German scientific communities did not become professionalized until later in the 19th century. Collab-

oration in the early decades of the 19th century was found to be mainly limited to French scientists, particularly the elite. Nearly 75 percent of collaborative papers sampled were produced by two or more French authors. In the latter portion of the 19th century, when the British and the German scientific communities became more professionalized, their share of collaborative papers rose accordingly.

Beaver and Rosen also show that collaboration increases the productivity of collaborating scientists, where productivity is measured in terms of the number of papers published. For younger scientists, collaboration, especially with the scientific elite, offers increased access to research facilities (both information and equipment) and increased visibility within the scientific community, the latter being especially important in a professionalized community. The increased productivity resulting from collaboration also makes it easier for the scientific community to justify support from outside society.

Whatever the causes of the increasing amount of collaboration in scientific research, two facts are clear: both the amount of collaborative research and the number of collaboratively produced research papers are increasing. This provides motivation for developing computer-based support for group-writing.

In the future, the trend towards increasing amounts of scientific collaboration will be supported by high-speed computer networks [Gre90, LU89, Sti90], operating at speeds over one gigabit per second and employing fibre optic technology [Sti90]. These networks are designed to support collaboration among scientists by allowing them to work with remote equipment and colleagues as if they were colocated. The tools these networks provide to achieve this will include digital libraries, multi-media electronic mail, group-writing tools, and multi-media teleconferencing. Advances

in data compression [CW84, Mof90] and in variable rate encoding schemes [CD89, Gha89, KT86, VG90, WCL<sup>+</sup>90] will further enhance the ability of these networks to transmit large amounts of data.

Indeed, facilities such as remote login and the transference of large amounts of data between remote computing sites are available on current computer networks such as the Internet, providing many opportunities for scientists to collaborate [SW90]. These capabilities have been exploited, at least to some extent: the use of large data banks functioning as extensive catalogues, such as data banks containing references to scientific literature (DIALOG, MEDLINE, etc.), is widespread; transferring data files and documents over such networks is also common.

### **How Scientists Collaborate**

Kraut and Egidio [KE88] have examined the effect of physical proximity on the development of collaborative relationships between scientific researchers and on the execution of their work. They found that there is a strong relationship between physical proximity and the likelihood of any two scientists collaborating; scientists whose offices are closer together are more likely to be collaborators.

Two explanations are offered for this behavior. First, scientists with similar research interests are more likely to have their offices colocated, and are more likely to collaborate in any event. For example, in a university members of the same department usually have their offices in the same area of a building. Second, researchers with offices near each other can easily communicate. This allows potential collaborators to assess each others' capabilities, and to manage their work more efficiently.

### 2.1.2 Collaboration in Business

Thus far the discussion has focused mainly on collaborative writing within an academic setting, but this practice is also increasingly common in business, in industry, and in government agencies [Str89]. It is common for many employees to collaborate on the production of a document, either as part of a group-writing team, or through delegation and ghostwriting [Cro90]. Common examples include the production of software documentation [Dal87] and the production of the executive letter of an annual report [Cro90]. Lunsford and Ede [LE86] (cited in [Str89]) surveyed 1200 professionals in management, engineering, chemistry, communications, and behavioral science, and 87 percent of the 530 respondents reported that they sometimes participated in group writing.

## 2.2 The Writing Process

During the past decade, cognitive models of writing have generated much interest [Nys86, SP88]. Most cognitive models view writing as an under-constrained goal oriented activity [SP88], and highlight the central role of writer intention, purpose, plans, etc. in the composing process [Nys86]. Representative of these models is the one developed by Flower and Hayes [FH81] (cited in [Nys86]), shown in Figure 2.1.

In this model the writing process consists of three main operations: *planning*, *translating*, and *reviewing* [HF80a]. Additionally, these operations are overseen by a *monitor* process, which determines when control passes from one operation to another. Planning, which consists of three suboperations, *generating*, *organizing*, and *goal setting*, takes information from the task environment and from long-term



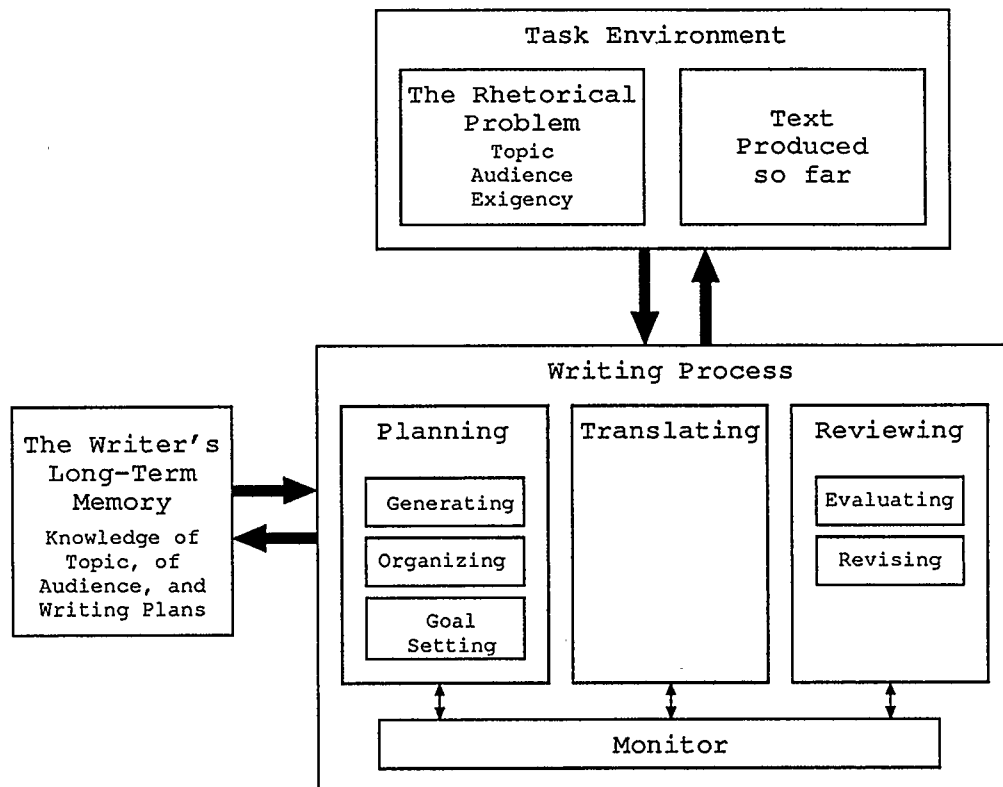


Figure 2.1: The Flower and Hayes Model of the Writing Process

memory and uses it to guide text production. Translating uses the writing plan together with long-term memory to produce text. Reviewing, which consists of the suboperations *evaluating* and *revising*, aims to improve the quality of the finished text. These writing operations form a precedence hierarchy, with editing and generating being the most important [HF80b]. Sharples and Pemberton [SP88, SP90] have extended this cognitive model of writing to incorporate the medium upon which writing takes place.

The cognitive model of writing is not without criticism. First, it does not account for how the situation in which the writer operates may shape the composing process [Flo89]. Second, it does not account for the social context within which writing takes place [Nys86]. Third, the principles organizing the hierarchy of cognitive processes are unclear, as are the principles organizing the generation of plans and of language [Nys86].

The use of word processors and computer-based document preparation systems has a major impact on the writing process. Word processors convey many advantages over pen and paper or typewriters when composing a document. It is much easier to make corrections and substantial changes to a document when using a word processor, because recopying of text is unnecessary. The existence of tools to check spelling and grammar also simplifies the writing process. Finally, the widespread use of electronic networks has made the exchange of documents much easier.

How does the use of a word processor affect the writing process? Dickinson [Dic86] found that the use of a computer in a classroom of school children increased the amount of peer collaboration. When using a computer instead of pen and paper in a collaborative writing session, children were more likely to express their thoughts to

other group members. This may be partly because only one computer was available to the collaborative groups under study, meaning that only one group member at a time could actually use the computer. Dickinson also postulates that use of a computer enables children to concentrate more on spelling and content rather than on penmanship.

Studies of the adult writers also show that word processors substantially influence the writing process. Haas [Haa89] found that when writers use word processors, significantly less high level or conceptual planning occurs than when using pen and paper, and significantly more word or sentence level planning [Haa89]. A study by Bridwell-Bowles [BBJB87] corroborates these results. She found that word processors are better suited to writers who do most of their planning before beginning to draft a document. Those writers who employ constant rereading and planning as they write find adapting to word processors much more difficult. Possibly they suffer from not being able to see much of their writing at one time (as it is not currently displayed on the screen).

Further research is needed to gauge the effect on the quality of a finished document of the reduced planning that occurs when using a word processor. Most researchers agree, however, that planning is important and that good writers spend more time planning than average writers [Haa89]. Planning allows writers to explore, develop, and organize ideas, and facilitates comparison of different structures and approaches. Further, good writers are able to plan at higher conceptual levels [Haa89]. Some researchers believe that most of the major ideas and elements of a text are generated in the planning phase before writing begins [Haa89]. These factors suggest that the reduced amount of high level planning that occurs when using

a word processor may decrease the overall clarity of a finished document. Interestingly, related results have been found in computer-aided engineering. A study by Murotake and Allen [Wol91] found that increased use of computer tools during the conceptual phase of a project resulted in significantly less innovativeness in solving problems.

Word processors also affect the amount of revision done by writers. Daiute [Dai86] studied the use of word processing programs by junior high school students. She found that when revising text, students had more word and sentence level revisions than when revising with pen and paper, but fewer revisions overall. This is important, because Somers [Som80] has found that experienced writers tend to make more high-level and more structural revisions than less experienced writers.

Another problem with word processors is that writers often have difficulty in getting a sense of “where they are” in the text, they find it difficult to read critically, and have trouble moving to a specific location in the text [Haa89]. These problems are particularly acute when computers with low resolution or with small screens are used [HH86]. The use of large, high resolution screens capable of displaying as much text as a page of paper, and of displaying text in its final form, does much to alleviate this problem [HH86].

Philosopher Michael Heim [Hei87] has examined both traditional writing and writing with a word processor. He argues that while traditional writing methods encourage linear thinking and contemplative awareness, writing produced with a word processor “will be probably less intelligent, less carefully formulated, less thoughtful text. Computers may boost productivity...only to have a greater proportion of written stupidity” (p. 129 of [Hei87]). Writing will be less reflective, less carefully

formulated, because there is less difficulty in getting started, and the physical act of writing is much easier. This may also result in less directed text, because the ease of writing makes it easier for writers to explore possibilities. Finally, the greater potential for text production may lead to stress from the constant drive to produce and to control.

Stratton [Str89] has proposed three models of group-writing: the *horizontal division model*, the *sequential model*, and the *stratification model*. In the horizontal division model, each group member is responsible for producing one or more sections of the final document (Figure 2.2), including all the necessary researching, planning, drafting, and editing. This method of group-writing can lead to inconsistent terminology, inconsistent style, and duplication of effort. Nonetheless, 24 percent of the professionals who responded to a survey by Lunsford and Ede [Str89] reported using this method frequently.

In the sequential model, one group member produces a draft of the document and passes it to another group member, typically an editor (Figure 2.3). The editor then reorganizes the document, rewrites it, revises it, and reedits it, before passing it along to another group member, typically a publications manager. The publications manager repeats the process. If the actions of the editor and the publications manager have introduced inaccuracies into the document, the whole process may have to be repeated one or more times. This method, often used within the academic community, has the advantage of producing a consistent document. However, it also has the potential to cause divisions within the group, with each member regarding the others in the sequence as a “barrier to efficient and effective communication” [Str89].

In the stratification model, a group-writing team typically has a project manager,

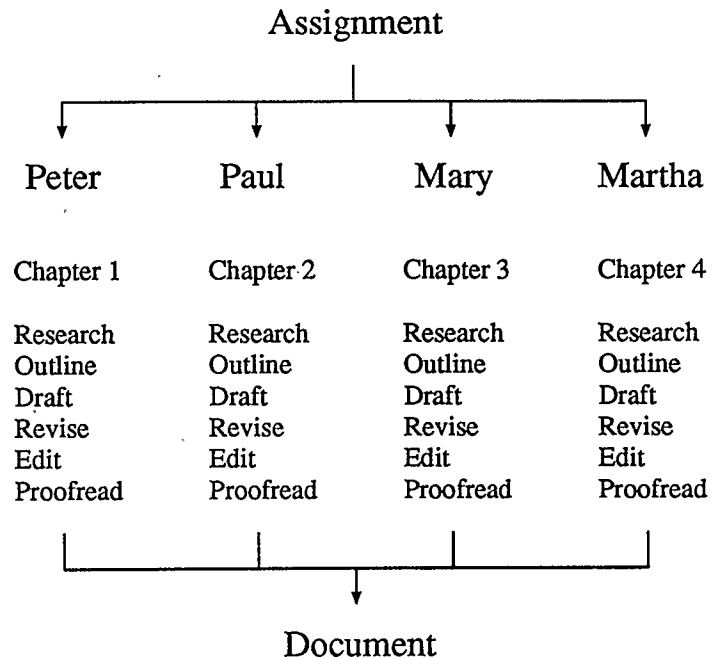


Figure 2.2: The Horizontal Division Model of Group-Writing

a data gatherer, a writer, an editor, and a layout designer (Figure 2.4), though the number of people assigned to each of these roles varies depending on the group size. Group members have non-overlapping areas of responsibility, and work together from the beginning to produce the final document. This model has the advantage that each member may work in their area of expertise and of interest, and is often used in the business community when producing technical documentation or reports. It encourages group members to work together instead of against one another, and the project manager can coordinate their effort.



Assignment



Peter

Research  
Outline  
Draft  
Revise  
Edit  
Proofread



Paul

Reoutline  
Redraft  
Re-revise  
Reedit  
Reproofread



Mary

Re-re-revise  
Re-reedit  
Re-reproofread



Document

Figure 2.3: The Sequential Model of Group-Writing

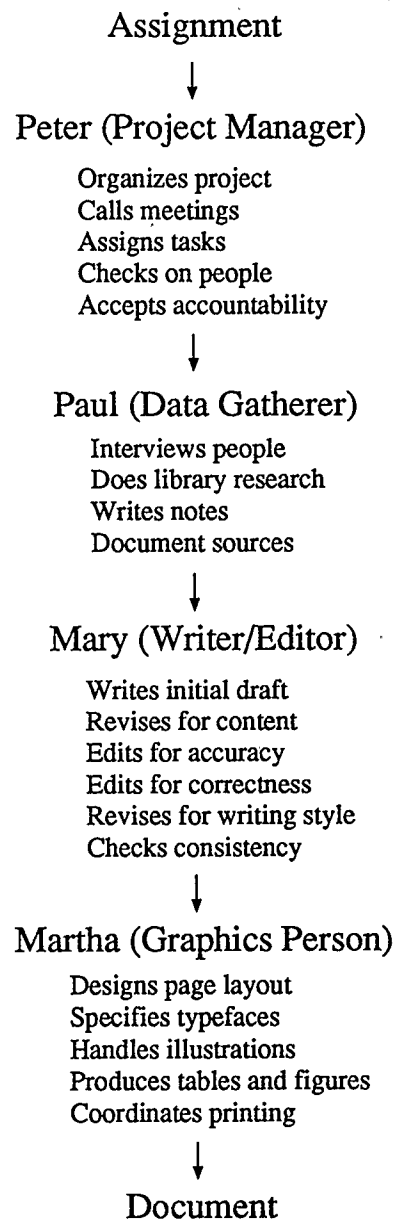


Figure 2.4: The Stratification Model of Group-Writing

### 2.3 General Requirements for a Group-Writing System

The previous sections of this chapter have surveyed some of the background behind group writing. Collaborative writing in science and industry, and aspects of the writing process were both considered. Visible against this background are some general requirements to be satisfied by a group-writing system in order for it to be widely used.

For groups using the horizontal division model or the stratification model, or for groups containing geographically dispersed members, allowing parallel versions of a document is important. In the stratification model, it is common for the writer to circulate a draft among other group members, e.g. the data gatherers, to collect their comments regarding the accuracy or style of the document.

Maintaining the different versions of a document as it evolves over time would also be useful, especially for groups using the sequential or the stratification models of group writing. Retaining document versions would also record the flow of ideas and allow the group to revert easily to a previous version if later changes are deemed inadequate. The latter point is especially important for professional writers, who currently often keep paper copies of all previous versions of a document.

If multiple versions of a document are stored, then a means to compare them easily should also be provided. This would allow a writer to easily examine changes made by others.

Increases in the use of computer networks, and in the amount of collaboration between geographically dispersed writers, highlight three especial concerns: compatibility, privacy, and reliability. Writers collaborating at a distance may often be

using different word processors, hence it is important that a group-writing tool be compatible with popular existing word processors. In many disciplines, loss or corruption of material can have serious legal or financial consequences. This is particularly true in areas such as law or biotechnology. Hence protection of privacy, and reliability in the face of network failure, are important issues.

Given the complexity of the writing process, it is important that a group-writing tool be simple and natural to use, so that users can concentrate on writing and not on the medium being used. Likewise, to accommodate writers with little computing experience, there should be provision for some members of a group to use a group-writing tool without having to manage multiple versions of a document.

To facilitate communication between group members, both annotation and electronic mail should be supported. Annotation allows users to record comments or to suggest revisions to a portion of text, and may be read at a later stage by other users. Additionally, these annotations should not disrupt the text of the original document version.

## 2.4 Specific Requirements for a Group-Writing System

In this section some specific requirements for the design of a group-writing system are proposed:

1. *Reproduction of older versions.* It must be possible to trace the evolution of a document, and to revert to a previous version if necessary [LH88].
2. *Maintaining multiple parallel versions of a document.* The existence of multiple parallel versions of a document must be allowed. For example, several users

may be simultaneously editing the document, or there may be two different versions of a scientific paper, one for a conference and one for a journal.

3. *Comparison of versions.* The option of comparing alternative or previous versions of a document should be supported, highlighting differences between the text of different versions [Dal87].
4. *Annotation of document versions.* Comments and suggested revisions of portions of text made by members of a group-writing team should be recorded and easily accessible [Dal87]. Additionally, these annotations should not disrupt the text of the original document.
5. *Compatibility with other editors.* It must be possible to exchange files with other commonly used word processors.
6. *Simple and natural to use.* Owing to the complexity of the writing process, and the nontechnical background of many potential users, an acceptable group-writing system should be very simple and natural to use. Allowance should be made for expectations created by single-user systems such as Microsoft Word.
7. *No requirement to manage complexity.* Users should not be required to understand and manage the versioning system. Provision should be made for naive users to use the system as a conventional word processor.
8. *Support of electronic mail.* Electronic mail facilities should be integrated with a group-writing system to facilitate communication between writers.

9. *Reliably store information.* Changes to a document must be reliably stored despite computer or network failure [LH88].
10. *Protection of privacy.* Access by unauthorized people must be prevented, because loss or corruption of material may have financial or legal consequences [LH88].

These requirements are not intended to be the standard against which all group-writing systems pass or fail. They are intended instead to provide a reasonable basis for the design of a group-writing system.

Although it is a prototype system, GroupWriter attempts to satisfy most of the above requirements. It should be noted however, that GroupWriter is not intended to aid in initiating a group-writing project, but to facilitate its progress once it is underway. Galegher [GK90] has shown that computer-supported communication may be better suited for coordinating already existing collaborations than for starting new ones.



## Chapter 3

### Related Systems

This chapter examines previous computer-based systems that have been developed to support document preparation. Single-user systems are surveyed first, followed by several previous group-writing systems.

#### 3.1 Computer Support of the Writing Process

Computerized document preparation systems were initially developed in the early 1960s. In these early systems, a document was first created using a simple text editor (Figure 3.1) before being passed to a formatting program that was responsible for producing a formatted version of the document suitable for display on an output device such as a printer [Fur89]. Interspersed with the document text were formatting commands that specified the final layout of the document [Jol89]. For example, to center  $n$  lines of text with the NROFF formatter, the following command is used:

```
.ce n
```

The commands accepted by early formatters such as RUNOFF and ROFF were simple and low-level [AFQ89], due partly to the primitive nature of printers available at that time.

Document formatters were then enhanced by the addition of *macro commands* combining several low-level commands [AFQ89]. This made it easier to produce documents with a more sophisticated layout. The availability of more advanced printers

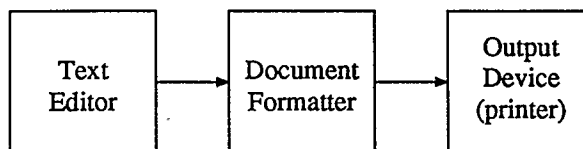


Figure 3.1: Document Preparation with a Pure Formatter

and of phototypesetters also led to more powerful commands, as in formatters like TROFF. Modern formatters, such as  $\text{\TeX}$  [Knu84], now handle page numbering, cross-referencing, compilation of indexes, contents tables, figures, tables, and mathematical formulae [Qui89], and can produce typographic quality documents.

The formatters discussed so far accept commands describing the physical appearance of the finished document. An alternative, developed in the 1970s, is to provide a logical description of the document—to describe the document in terms of chapters, of headings, etc. [AFQ89] The formatter then derives the page layout from the logical description of the document. One of the most influential systems of this type was Scribe [Rei80], developed by Reid at Carnegie-Mellon University. The heading “Computer Support of the Writing Process” is expressed as

```
@Heading(Computer Support of the Writing Process)
```

when using Scribe. Another important system of this type is  $\text{\LaTeX}$  [Lam86].  $\text{\LaTeX}$  attempts to provide the functionality of Scribe while building on the typographic quality and the ability to handle mathematical formulae provided by  $\text{\TeX}$ .

When documents are described in terms of their logical components, and not their final appearance, the document description is independent of particular computers and output devices. This offers many advantages. Documents can be shared between users without concern for compatibility of printing devices [CRD87]. The publi-

cation cycle is made faster and more efficient [CRD87]. Online databases for electronic publishing are made more feasible [CRD87], as is electronic interchange of text [Bry88]. Finally, writers are free to focus on the structure and the contents of a document, without regard for physical presentation [CRD87].

Document preparation systems designed to operate on a logical description of a document can potentially offer many powerful services [CRD87], because the different logical components of a document can be easily identified. For example, *alternative views* of a document can be offered. The editor can view all elements of a certain type, e.g., headings, or filter all elements of a certain type from view, e.g., footnotes. *Outlining* capabilities can be offered. Editing commands can be more powerful, e.g., to copy or delete an entire section of a document.

Recognizing the advantages offered by describing documents logically, in 1986 the International Standards Organization (ISO) introduced Standard ISO 8879. This standard is for a Standard Generalized Markup Language (SGML) which can be used to describe a wide variety of documents without regard for the input or the output devices involved [Bry88]. SGML was developed by Goldfarb, and is largely based on IBM's Generalized Markup Language (GML) [Bry88].

With the advent of high resolution displays and graphical user interfaces, integrated editor/formatters have become much more common. The editing and formatting functions are merged in an integrated editor/formatter [Fur89]. Creation, viewing, and reviewing of a document is done without ever having to leave the system (contrast this with Figure 3.1, where a separate editor and formatter are required). Common editor/formatters are What You See Is What You Get (WYSIWYG) systems like Microsoft Word. WYSIWYG editor/formatters must have a fast response

time because of their interactive nature. Hence, they typically produce documents of poorer typographic quality, especially for word breaks, construction of lines, and construction of paragraphs, than pure formatters such as  $\text{\TeX}$  [Qui89].

Recently, interactive editor/formatters which utilize a logical document description have been developed [Qui89]. Examples include Interleaf [Mor85], Etude [H<sup>+</sup>81], Lara [Gut85], Speed [CIV86], and Grif [QV86]. Except for Interleaf, these systems are all experimental. Nonetheless, great progress can be expected in this area during the next few years.

Another recent innovation has been the development of computer systems for group-writing. These systems, which typically concentrate on annotation and on version management of documents, are now discussed in detail.

## 3.2 Group-Writing Tools

Providing computer support for group writing is currently generating much interest. There are already commercial systems that allow annotation of documents [Dal87], and many research projects. Contemporary systems fall into two classes: linear (non-hypertext) systems and hypertext systems (Table 3.1).

### 3.2.1 Linear Systems

This section concentrates on group-writing systems that are not based on hypertext.

#### **CES: The Collaborative Editing System**

The Collaborative Editing System (CES) [GS87, Sel85] is a group-writing system developed in the programming language Argus [LS83] at the Massachusetts Institute

Hypertext based	Non-hypertext based
Contexts	CES
ForComment	DistEdit
GroupWriter	GROVE
InterNote	In-Synch
PREP	MarkUp
Quilt	RCS
	Red Pencil
	Shared Books
	vmacs

Table 3.1: Group Writing Systems

of Technology. Each document in the CES system is explicitly composed of a hierarchy of sections (similar to a table of contents). The structure is used to provide a framework for coordinating author's activities; different authors may work concurrently on different sections.

A document in the CES system is composed of two types of objects: a *structure object* which stores the structure of the document, and one or more *node objects* which store a portion of the contents of the document. There is also a special type of node object called a *critique node*, which is used for annotation.

A user is presented with three windows—one giving the document outline, one to edit a section of the document, and one giving a broader context to the section of the document being edited. The document outline corresponds to the structure object, and there is one node object for each element in the document outline. The document outline is hierarchical, and includes critique nodes unless the user wishes them filtered from view. Users may simultaneously work on different sections of the same document. They are implicitly granted a lock when they begin editing a

section, which prevents other users from editing the same section. Changes made to a CES document can be saved on a stack, and recalled at any time.

Each document has a set of authors and associated access privileges (no-access, view-only, or edit); separate privileges for each section are not supported. To improve response time, each section of the document is stored at the workstation of its creating author, although it is still accessible to other users. The exception to this locality principle is the document outline, which is replicated among the workstations in the system.

### Shared Books

Shared Books [LH88] is a collaborative publication management system developed for use within the Xerox ViewPoint document processing system. A ViewPoint installation consists of a workstation environment with file, print, and communication services, in which each workstation has its own local storage. Sharing of information is done via file servers.

A Shared Books document is represented on a user's workstation as a window resembling a table of contents. Each entry in the table of contents is implemented by a file, and may be one of two types: a *body entry* which stores part of the content of the publication, and an *auxiliary entry* which holds additional information, e.g., correspondence.

Each entry in the window is displayed on a single line, containing the following information:

- The name and the class of the entry.
- The lock status of the entry—whether the entry is locked by the user, by

someone else, or not locked. Locking an entry provides exclusive access for editing.

- The revision number of the entry, which is incremented each time the entry is saved after major editing changes.
- The creation date of the most recently saved version of the entry.
- The “notes” field of the entry, which contains comments entered by a user.

Access controls can be set both for the Shared Book itself and also for each of its component entries. The use of access lists and user identities supports different roles for different users. A user may have several different identities (e.g., author, reader) each with different access rights to a Shared Book and to its entries.

Shared Books is similar in many ways to Seliger’s Collaborative Editing System. It allows a group of authors to work simultaneously on a document, and each author may display the structure of the document in a way that resembles a table of contents. Unlike the Collaborative Editing System, Shared Books provides revision and status information for each of the entries, and allows privileges to be set separately for each section of a document.

### **The Revision Control System**

The Revision Control System (RCS) [Tic82] was developed to manage multiple revisions of text files, particularly programs and documentation, and is similar to the Source Code Control System (SCCS) [Roc75]. RCS uses an archive file to record all changes made to a document. When a user wishes to change a document version,

they *check-out* the appropriate version from the archive file. When the changes are complete, the user must *check-in* the modified version, and RCS records any changes.

When storing a new document version, only the version deltas, or the differences between versions, are actually stored. From the end user's point of view, however, complete versions are stored. RCS actually stores the newest version intact; other versions are reconstructed by using deltas. This means that checking out the newest version for modification is fast, and is independent of the total number of versions.

To prevent two or more users logging competing changes to the same version, RCS uses locking. RCS locks any version checked out by a user, preventing any other user from checking in changes to the same version unless the lock is explicitly overridden. Branches in the version history of a document are allowed, and facilities are provided to merge different versions. Consider Figure 3.2: the initial document version is 1.1, and there are currently three branches. To merge version 1.2.1.1 with version 1.3, version 1.2.1.1 is retrieved and all changes leading from version 1.2 to version 1.3 are incorporated into version 1.2.1.1. If there are overlaps in the changes, the user is notified and RCS marks the affected areas. The user must then edit the file and delete those sections that are not desired.

For every version in the archive file, RCS maintains the following attributes: a revision number, the check-in time and the check-in date, the author's identification, a log message, and the actual text. An access list is maintained that lists all users who may access an RCS archive.

### Other Linear Systems

Other linear systems not considered in detail here include MarkUp [Mai], vmacs



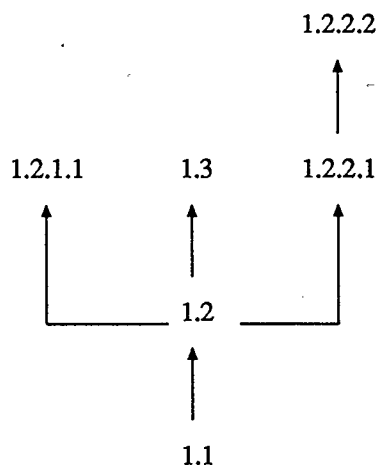


Figure 3.2: An RCS Version Tree

[Lak90], Red Pencil [Dal87], GROVE [EGR91], In-Synch [Dal87], and DistEdit [KP90]. MarkUp, vmacs, and Red Pencil allow reviewers to overlay annotations on a copy of a document, MarkUp using a bitmap copy of the document. GROVE and In-Synch support tightly coupled interaction between users, similar to some computer conferencing systems. DistEdit is a toolkit for modifying existing editors at the source code level, and also supports tightly coupled interaction.

### 3.2.2 Hypertext-based Systems

A hypertext document consists of a collection of nodes, typically containing text, graphics, or digitally encoded voice, connected by directed links [DS87]. These links allow hypertext<sup>1</sup> to be viewed as nonlinear or nonsequential text [Con87]. Links may be attached either to a specific region within a node, or to the whole node itself [DS87]. The set of all documents in a hypertext system is called a hyperdoc-

<sup>1</sup>Some researchers distinguish between *hypertext* and *hypermedia*. Hypermedia documents may contain text, graphics, digitally encoded voice, etc. Hypertext contains only text. This distinction between hypertext and hypermedia is not made here; the two terms are used interchangeably.

ument. Hypertext systems often provide *browsers* for displaying the relationships between the hypertext nodes. Browsers can be either hierarchical or graphical.

The concept of hypertext is generally credited to Vannevar Bush [Con87]. Bush wanted to make the ever-increasing amount of scientific literature manageable, and proposed the Memex, a mechanical device for storing books, etc., and with a fast retrieval time [Bus45]. The Memex is not a digital computer, but rather uses microfilm and photocells to store information.

One of the first computerized hypertext systems was developed by Engelbart [Con87] in the 1960s, at the Stanford Research Institute. Engelbart envisioned that the power of a hypertext system could be used to augment the human intellect. Another early hypertext visionary, Ted Nelson [Nel87], foresaw a future in which the entire world's literary corpus would be incorporated into a single, distributed, hyperdocument, in effect, a *docuverse*.

In spite of this enthusiasm, there are two major problems with hypertext systems [Con87]. Users tend to lose track of where they are in a hypertext document, and of how to find specific information. Using hypertext also incurs an additional cognitive overhead because hypertext documents are not linear.

In the development of computerized writing environments, hypertext has been a popular tool because it has facilities supporting several aspects of the writing process. First, hypertext naturally supports note-taking [DS89], individual ideas being stored on individual nodes. Second, links between idea nodes can be used to explore relationships between ideas during the planning phase of a writing project [Beg90, DS89]. To prevent information overload, some hypertext browsers can be tailored to filter out all links but those of a given type. Third, preparing a document outline is

supported in hypertext by facilities for grouping hypertext nodes and for organizing them into hierarchies [DS89]. In addition, some hypertext systems provide facilities for automatically maintaining references and bibliographies.

The ability of hypertext to support online annotation of documents [DS89] means that hypertext can easily support this aspect of group-writing. An annotation is formed simply by creating a node and linking it to the appropriate location in the document. Some hypertext environments, such as Intermedia, have been further enhanced to provide annotation capabilities specifically aimed at supporting collaboration [CBY89].

Generic hypertext systems, however, provide inadequate support for collaborative authorship of large documents [DS87]. Some of their shortcomings are listed below:

1. They do not provide independent hypertext partitions in which authors may work without the risk of interfering with each other, or facilities to join such partitions [DS87].
2. They do not allow version trees of hypertext nodes and links to be built [DS87].
3. They do not usually provide an easy way for *configurations* to be built. A configuration is a collection of specific versions of nodes and links [DS87].

Several hypertext-based group-writing systems are now discussed.

### Contexts

Contexts [DS87, DS89] extends the Neptune hypertext system, aiming to make it more suitable for collaborative document preparation. Neptune was designed to

support collaboration in engineering projects [DS89], hence it records all changes to nodes and links. Old versions of nodes or of links can be accessed at any time.

The Contexts system extends the Neptune system by allowing users their own private views or contexts of the hypertext database. A context consists of a collection of nodes and links. One context—the master context—is used to store the current draft of a document. Each user creates their own, separate, context for working on portions of the document. When the changes are complete, the user's context can be merged with the master view.

To prevent conflicts from arising when two or more authors simultaneously change the same portion of the hyperdocument, when one author checks out a portion of the document into their own private context, that portion of the document is set to read-only for other users. After any changes are complete, the merge operation provided in Neptune is used to install the new versions into the master view.

### Quilt

Quilt [FKL88, LFK88] is a computer-based tool for collaborative document production, providing structured hypertext facilities for either voice or text annotation of documents. Users may define their own annotation types using a table driven interface, or use one of the default Quilt annotation types listed below:

1. *Revision suggestions*, which contain alternative versions of a passage of text.

Users with appropriate permissions may swap a revision suggestion with the current version of the text.

2. *Public comments*, which may be read by anyone.

3. *Directed or private messages*, which are displayed only to named individuals and groups.

The annotation facilities allow discussion ideas to be attached directly to the relevant portions of text. In addition to annotation, Quilt also supports electronic mail. Quilt can send messages to the authors whenever substantial changes are made to a Quilt document, and reminder messages can be sent with increasing frequency as the deadline for completion of the document approaches.

To manage the complexities involved in group-writing, Quilt assigns different social roles to the partners in a collaboration. Typical roles are *reader*, *commenter*, and *co-author*, each having different permissions: read; read and annotate; read, write, and annotate. Users may define additional social roles via a table-driven interface.

Many hypertext systems provide a graphical browser, and Quilt is no exception; users can browse the general structure of a Quilt collaboration using a graphical representation of annotation links. As with Contexts [DS87, DS89], Quilt can store history versions of the document, complete with annotations.

Quilt was implemented using existing tools wherever possible. The underlying database system, which stores and retrieves information about users, collaborations, and documents, is implemented using Orion. Window management, including the display of information on the screen and response to user input, is handled by X and the XR toolkit.

Orion supports class hierarchies, and the base class for document components is the Quilt Object class, whose attributes include *type* and *creator*. *Type* is either

node or link. Node objects store text or voice information, and link objects join one node object to another. The `creator` attribute records the creator of the object.

Node objects have attributes for `data type`, `links in`, `links out`, and the data itself. The `data type` attribute describes the type of information and how to present it. The `links in` and `links out` attributes describe sets of links leading to and leading from the node. The data attribute stores voice or text information.

Link objects have attributes for `annotation type`, `from node`, and `to node`. The `annotation type` attribute describes the type of the link (e.g., directed message). The `from node` and the `to node` attributes are for the nodes at either end of the link. This structure permits arbitrary nesting, allowing annotations to be annotated.

## PREP

PREP [NKCM90], a “work in preparation” editor developed at Carnegie Mellon University, is designed to support planning, organized annotation, and communication between authors. It is currently implemented using MacApp on Macintosh II computers.

Nodes in the PREP system typically contain text or arbitrary images, and are stored in a database that is shared among collaborators. Nodes may be arranged into grids (matrices) or trees. When using PREP, authors define *drafts*, which are sparsely filled grids with an arbitrary number of columns. Typically there will be one column for the document plan, one for the document text, and another for annotations. Reviewers may add additional columns to hold their comments, and distinct versions of the document may be saved at any time.

PREP is an impressive system, which pays particular attention to the cognitive aspects of collaborative writing. More so than other systems, it allows co-authors and commenters access to planning information, and nested annotations.

### **Other Hypertext-based Systems**

Two other hypertext-based systems not considered in detail here are InterNote [CBY89] and ForComment [Dal87, Opp88]. InterNote is an extension of the Intermedia hypertext system, supporting annotative collaboration. ForComment is a PC-based tool which allows annotations to be connected to each line in a document.

### **3.2.3 A Comparison of Group-Writing Systems**

The hypertext and the non-hypertext group-writing systems discussed in detail above are now compared (Table 3.2) with respect to the quantifiable requirements in Chapter 2 for a group-writing system. In terms of these requirements, the linear system providing the best support for group-writing is RCS. However, the hypertext systems generally provide better support for group-writing, especially for history versions of a document, parallel versions, version comparison, and annotation.

	Hypertext Systems			Linear Systems		
	Contexts	Quilt	PREP	CES	Shared Books	RCS
History versions	y	y	y	y	?	y
Parallel versions	y	?	y	n	n	y
Version comparison	y	y	n	n	n	y
Annotation	y	y	y	y	y	n
Compatible with other editors	?	?	y	?	?	y
Some users need not manage versioning	n	y	y	y	y	y
Electronic mail	?	y	n	n	n	n

Table 3.2: A Comparison of Group-Writing Systems



## Chapter 4

### Document Storage

Commercial word processors typically do not maintain the version history of a document as it evolves over time. It is often desirable, however, for a writer to have access to earlier versions of a document. This would allow different document versions to be easily compared, and for changes that have proved unacceptable to be easily discarded. The ability to recover from changes is especially important when several writers are collaborating on a document.

Another desirable feature is to allow writers to work concurrently on the same document, even if they are at geographically dispersed locations. This creates, however, many problems. As in distributed databases [JC85] and in shared-memory multiprocessors [Sto87], concurrent access to shared data must be carefully controlled, otherwise the integrity of the data could be destroyed. In addition, writers must be prevented from accidentally destroying changes made by others.

The solution adopted in GroupWriter is to use *immutable* document versions. Immutability is the property of changelessness, unalterability. An immutable document version cannot be changed—each time it is modified, a new document version is created.

Immutability has been used in distributed file systems, such as the Bullet file server [vRTW89] and the Cedar File System [GNS88], and in software engineering environments, such as RCS [Tic82] and the Cosmos Distributed Programming Environment [BMNS87, NBW88, WBMN88]. It is simple to implement and, in a

distributed environment, relieves concern about whether all copies of an object are consistent. All copies of an immutable object are, by definition, identical.

Immutability introduces a new problem: version management. The most simple solution, adopted in the Cedar File System, is to maintain only the  $n$  most recent versions of an object; if a new version of an object is created, and more than  $n$  versions are extant, the oldest will be deleted. This method has some drawbacks. It can result in the deletion of objects a user wished to keep. It also allows a single user to cause everybody else's changes to shared data to disappear [GS87].

RCS avoids this problem by retaining all versions of a document, unless specifically requested otherwise. RCS also allows branches in the version history of a document (Figure 3.2), although they must be explicitly created. One branch in the version history of a document could be for a conference paper, another for a journal.

An even more general approach towards branching is used in the Cosmos system [WBMN88] by *free branching objects*. If two or more users change the same free branching object, they automatically create their own new versions of the object and a branch in the version history of the object (Figure 4.1).

GroupWriter uses the Cosmos solution. If two or more users simultaneously edit the same version of a document, then a branch is automatically created in the version history. Furthermore, GroupWriter stores all versions of a document—none are ever deleted.

A potential problem in the use of immutable document versions is inefficient use of storage. Storing many versions of an object as opposed to only the most recent can incur a large storage overhead. In storing immutable files, Cedar [GNS88] adopts the simplest approach of maintaining a complete copy of each version. RCS [Tic82],

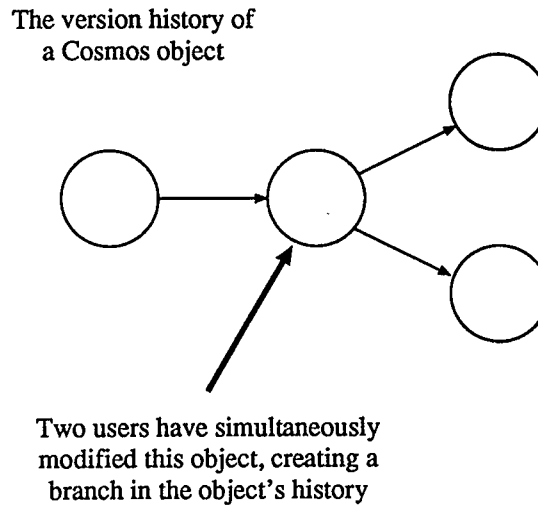


Figure 4.1: Free Branching Objects in Cosmos

on the other hand, only stores the differences between document revisions. Thus, if only small changes are made to a file, RCS uses storage space much more efficiently than Cedar.

Like RCS, GroupWriter stores the differences between document versions. When a document version is saved, only modified paragraphs or newly created paragraphs or annotations are stored. This contrasts with RCS, which stores differing lines, not paragraphs. The storage scheme used in GroupWriter is discussed in depth in Chapter 6, and an empirical comparison with RCS is contained in Chapter 7.

## Chapter 5

### Overview of GroupWriter

This chapter provides an overview of the use of GroupWriter. The first section details the assumptions behind the design of GroupWriter, and provides a brief description of GroupWriter's features. The second section provides a detailed description of the features provided by GroupWriter.

#### 5.1 GroupWriter in Brief

Users are assumed to have Apple Macintosh computers with enough local storage to hold at least one version of a document. Documents are managed and archived through a server on a network accessible to all users, and editing is done on a Macintosh using a local copy of a document version. Users need only use the network intermittently, to obtain other versions of a document and to commit any newly created versions back to the server, making them available to others.

GroupWriter automatically retains previous versions of a document, which may be accessed at any time. Additionally, multiple parallel versions of a document may exist, i.e., there may be branches in the document history. Branches arise under two conditions. First, if two or more users simultaneously edit the same document version a branch is created in the document history. Second, users can explicitly create a branch in the document history, for example, to create a different version of a paper.

Merging different versions of a document is also supported. Users can examine alternative versions of paragraphs in a variety of ways and can choose which they wish to insert into the text of the merged document version. GroupWriter is a hypertext-based system, allowing annotations to be attached to individual paragraphs and inspected at will. History versions of a document maintain a complete record of all alternative versions of paragraphs, and of all annotations. Relationships between document versions can be displayed and manipulated graphically.

In maintaining the version history of a document, two files are used. The *archive file* stores all the text and annotations occurring in all the versions. The *structure file* stores all of the information necessary to reconstruct, from the data in the archive file, any of the document versions.

## 5.2 GroupWriter in Detail

This section describes in detail the operation of GroupWriter, emphasizing how its main features are used.

### 5.2.1 Basic Editing Operations

GroupWriter is designed to be simple and easy to use. Document versions can be created, edited, and saved as in a normal word processor on the Macintosh (e.g. Microsoft Word). To begin a new collaborative document, the user selects **New** from the **File** menu, and an empty window appears. After editing is complete, this first document version is saved by selecting the **Save** option from the **File** menu. GroupWriter will prompt the user for the names of the structure file, the archive file,

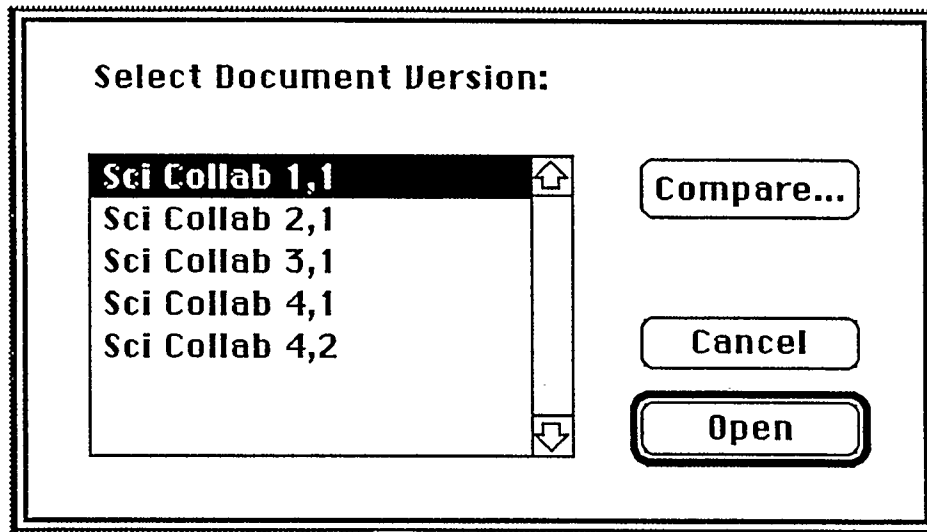


Figure 5.1: Opening a Document Version

and the document version. The structure and the archive files must be stored in the same directory, so that they can both be found when opening a document version for editing.

To edit an existing document version, **Open** is selected from the **File** menu. A standard file opening dialog is displayed, allowing selection of the desired structure file. Once the structure file has been chosen, a dialog appears listing all the document versions in the structure file (Figure 5.1). Document versions are listed in the order of their creation. After selecting the appropriate version and clicking the **Open** button, the desired version appears in a normal edit window. Once editing is completed, **Save** is selected from the **File** menu and the user is prompted for the name of the new document version.

By default, GroupWriter automatically numbers document versions according to their position in the document history tree. If a user wishes, this option can be

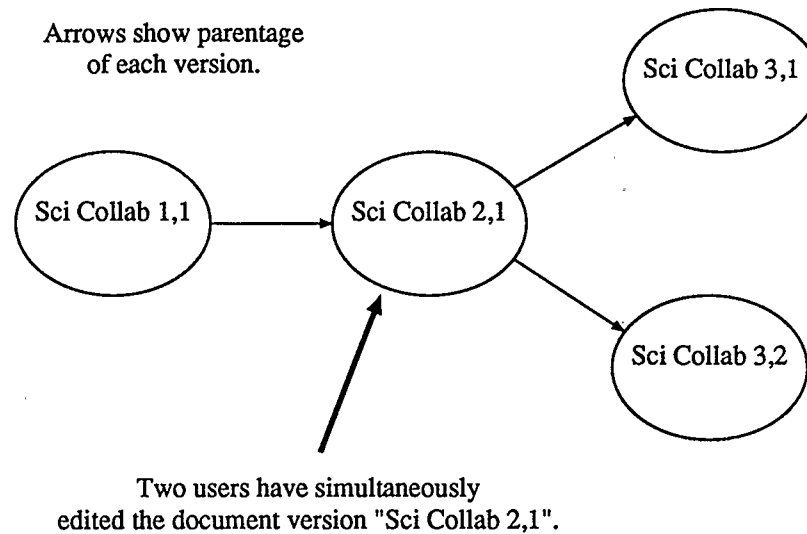


Figure 5.2: Branches in the Version History

disabled. Versions names need not be distinct: it is always possible to distinguish between document versions by using the graphical version browser (see Section 5.2.4).

If two or more users simultaneously edit the same document version a branch is automatically created in the version history of the document (Figure 5.2). Unlike RCS [Tic82], no branch assumes priority over the others.

### 5.2.2 Comparing Different Document Versions

Like RCS and Quilt, GroupWriter provides facilities for comparing document versions with one another. The document version opened for editing is the *primary version*. GroupWriter allows the primary version to be compared with one or more other document versions. Document versions that are to be compared with the primary version are called *secondary versions*. When opening a document, the secondary versions are selected by clicking the Compare button (Figure 5.1). A dialog containing a list of the document versions is displayed, allowing secondary versions to be

selected.

GroupWriter then searches through all the secondary versions, finding any alternative versions of the paragraphs in the primary version (details of this process are in Chapter 6). Quilt also compares document versions on a paragraph basis, whereas RCS uses lines. The paragraph was chosen as the unit of comparison because it is a natural conceptual unit for writers [HH76, SW79].

When the comparison is complete, the primary version is loaded into a normal edit window and editing can begin. Paragraphs with an alternative version have a corresponding bracket in their left margin (Figure 5.3). Paragraph brackets are reasonably robust, being preserved through Cut, Copy, and Paste operations. The mouse pointer changes shape when moved over the lower portion of a paragraph bracket, indicating that a popup menu is available. Clicking the mouse causes the popup menu to be displayed (Figure 5.4).

The popup menu is hierarchical in structure and displays both the list of alternative paragraphs and the operations that can be performed on them. Alternative paragraphs are identified by the name of the document version in which they occur. The operations that can be performed on alternative paragraphs are listed below:

1. **Show.** Display the selected alternative paragraph in a separate, non-editable window. This window is especially positioned to keep the original paragraph in the main edit window visible.
2. **Replace.** The selected alternative paragraph replaces the paragraph in the main edit window. The replaced paragraph is inserted into the list of alternatives.



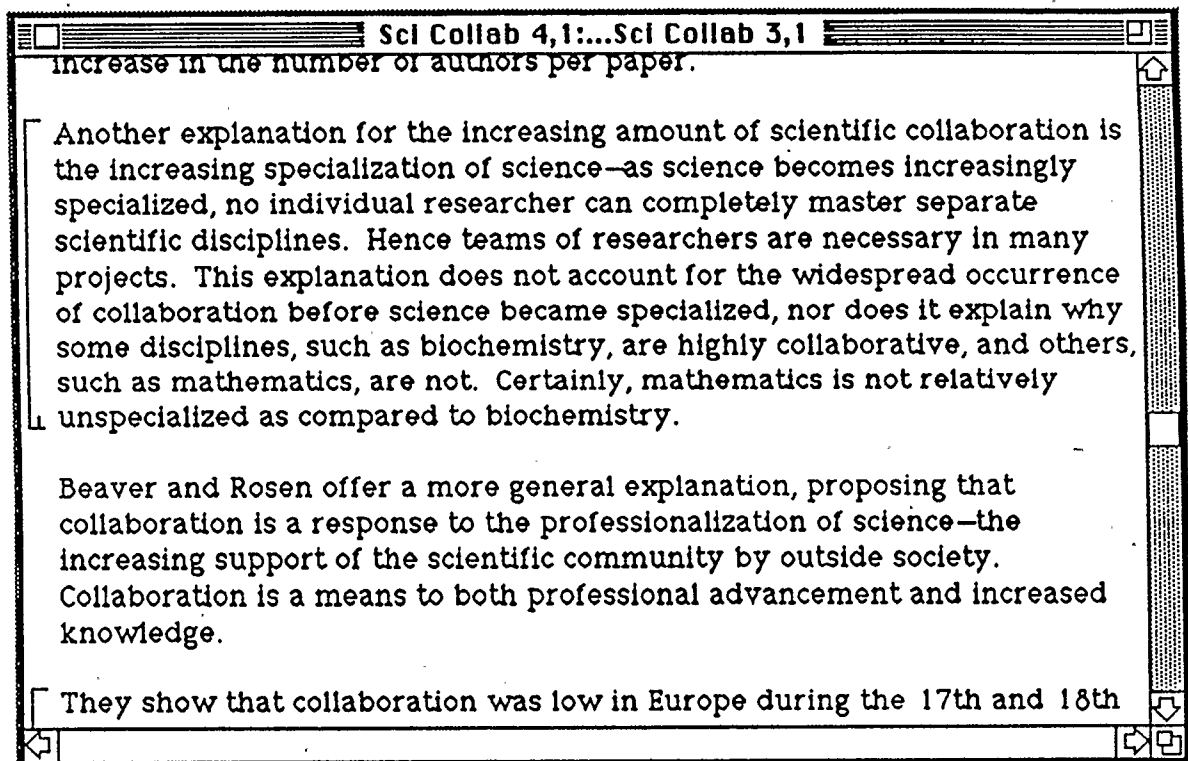


Figure 5.3: Paragraph Bracketing

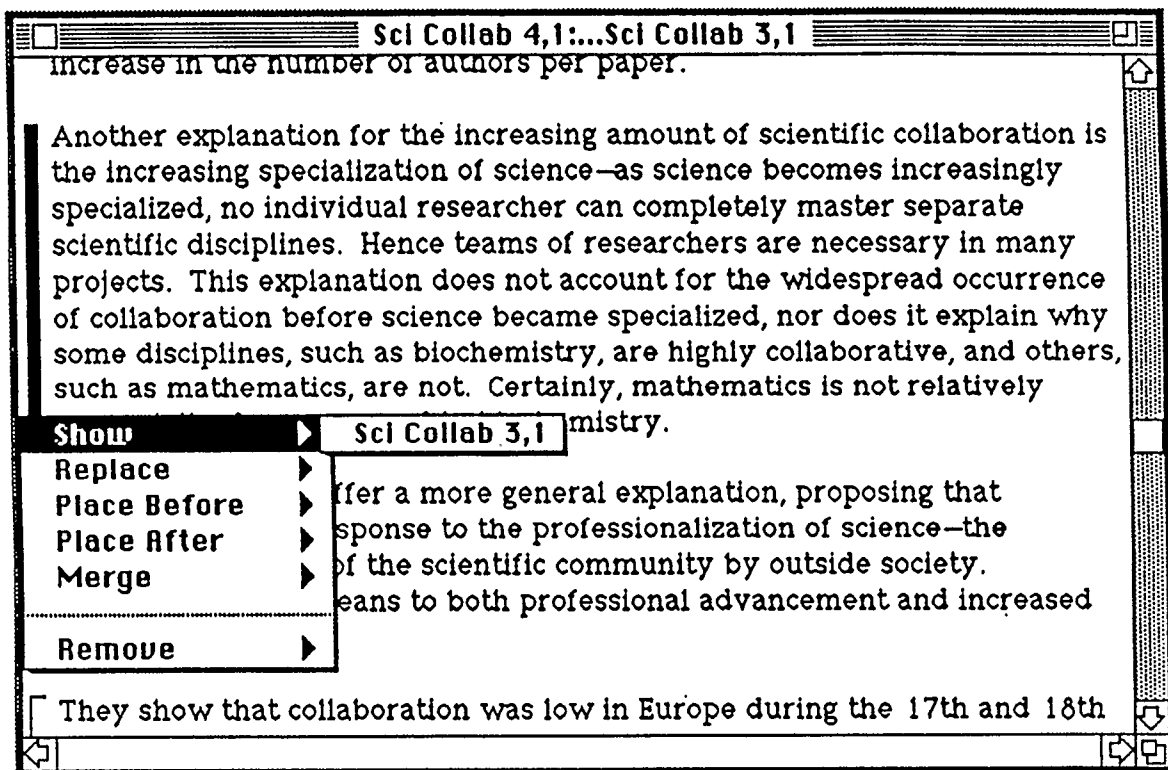


Figure 5.4: The Popup Menu and Alternative Paragraphs

3. **Place Before.** The text of the selected alternative paragraph is inserted before the paragraph in the main edit window.
4. **Place After.** The text of the selected alternative paragraph is inserted after the paragraph in the main edit window.
5. **Merge.** Compare the selected alternative paragraph with the paragraph in the main edit window on a sentence-by-sentence basis. Differences between paragraphs are displayed in a separate non-editable window (Figure 5.5). Sentences which differ between the paragraphs have a bracket in the left margin; sentences which are common to both paragraphs do not. The paragraphs can then be merged, replacing the original paragraph in the main edit window. Merging proceeds as follows. Sentences that are common to both paragraphs are automatically incorporated into the merged paragraph. Sentences that differ are included by selecting the corresponding bracket. More than one differing sentence can be included by holding down the <shift> key when the selection is made. The merge is completed by double-clicking in a bracket.
6. **Remove.** The selected alternative paragraph is removed from the list. As shown in Figure 5.4, this option is separated from the others in the popup menu by a dotted line, in order to reduce the chances of it being inadvertently selected.

The Replace, Place Before, Place After, and Merge operations cause the text in the main edit window to be changed. The changes are highlighted, to make them more visible.

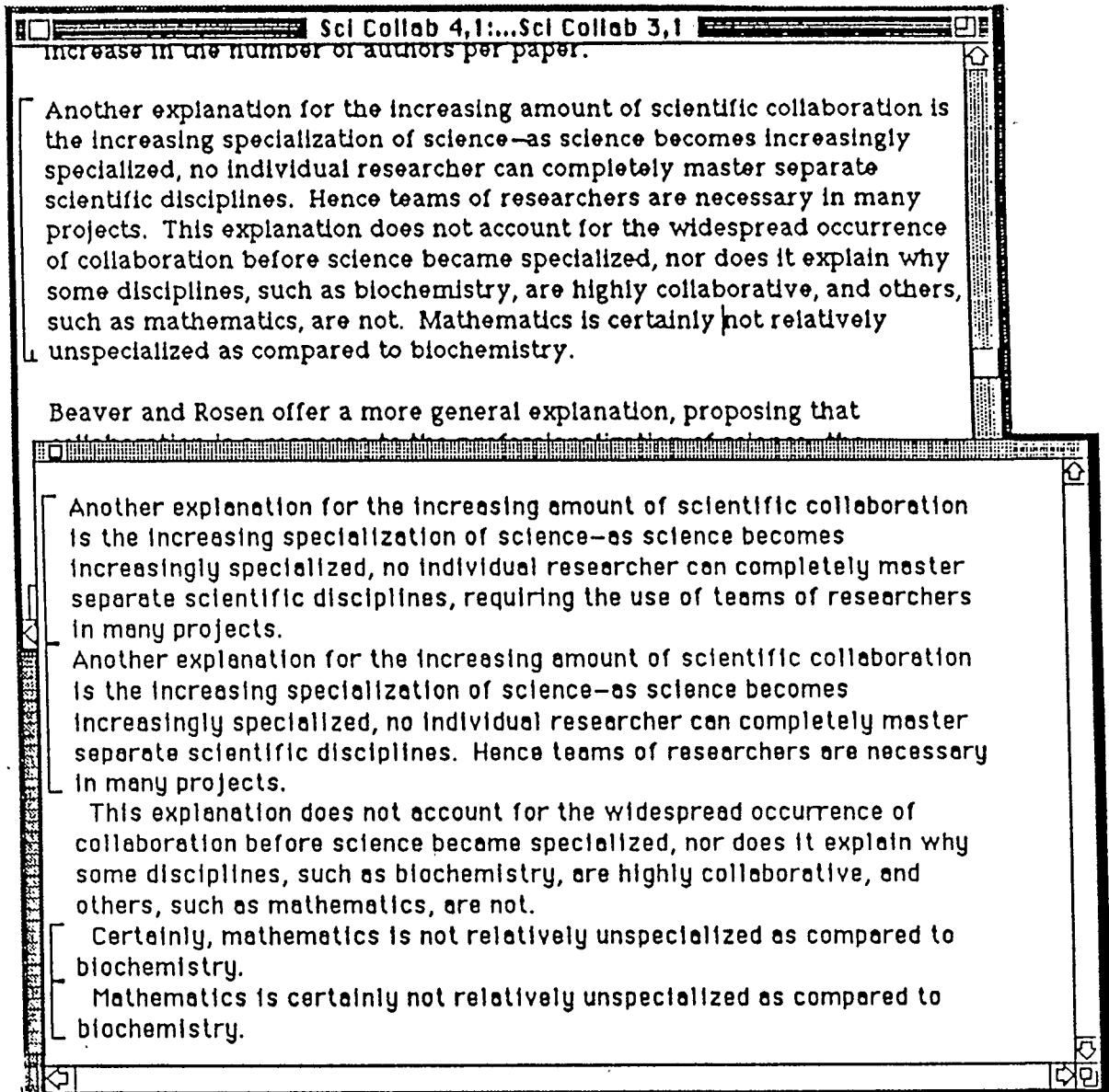


Figure 5.5: Viewing the Difference between two Paragraphs

### 5.2.3 Annotation Facilities

Similarly to other hypertext-based group-writing systems, GroupWriter supports the annotation of document text, though it provides this support at the paragraph level. A paragraph is annotated by first placing the cursor within the paragraph to be annotated, and then selecting **Annotate** from the **Edit** menu. This causes an annotation dialog to appear, with space for both the title and the text of the annotation (Figure 5.6). After the annotation is completed, the user selects the **OK** button. As with alternative paragraphs, the presense of annotations is indicated by a bracket in the left margin. Annotations are accessed via the same popup menu used for alternative paragraphs, though they are distinguished from alternative paragraphs by a dotted line on the menu (Figure 5.7). All the operations that can be performed on alternative versions of a paragraph can also be performed on annotations. When a document version is saved, enough information is stored in the structure file to reconstruct all of its annotations the next time it is opened.

### 5.2.4 The Graphical Browser

GroupWriter provides a graphical browser for examining relationships between document versions. Selecting the **Graph** option from the **File** menu causes a dialog to appear requesting the name of a document structure file. After choosing the desired structure file, a window containing a graphical view of the document versions appears (Figure 5.8).

Each arrow shows the parentage of a version. For example, **Sci Collab 3,1** is the parent of both **Sci Collab 4,1** and **Sci Collab 4,2**. The ovals representing document versions may be selected and dragged to different positions in the window.

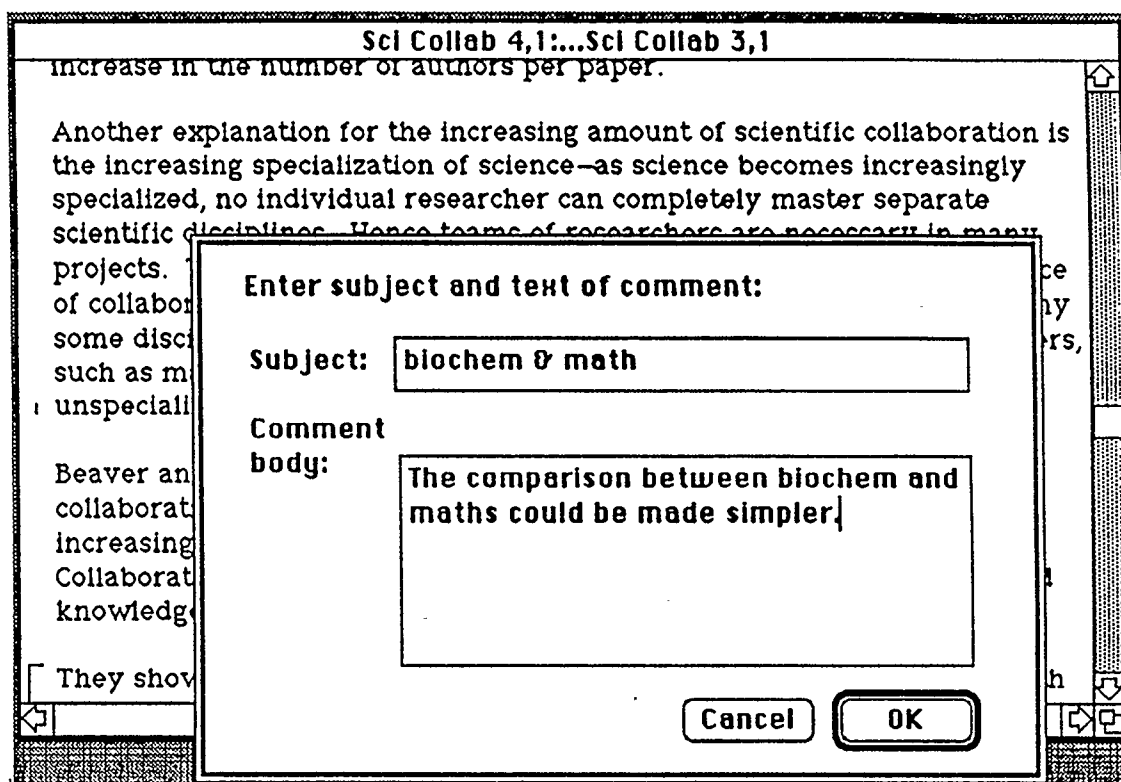


Figure 5.6: Annotating a Paragraph

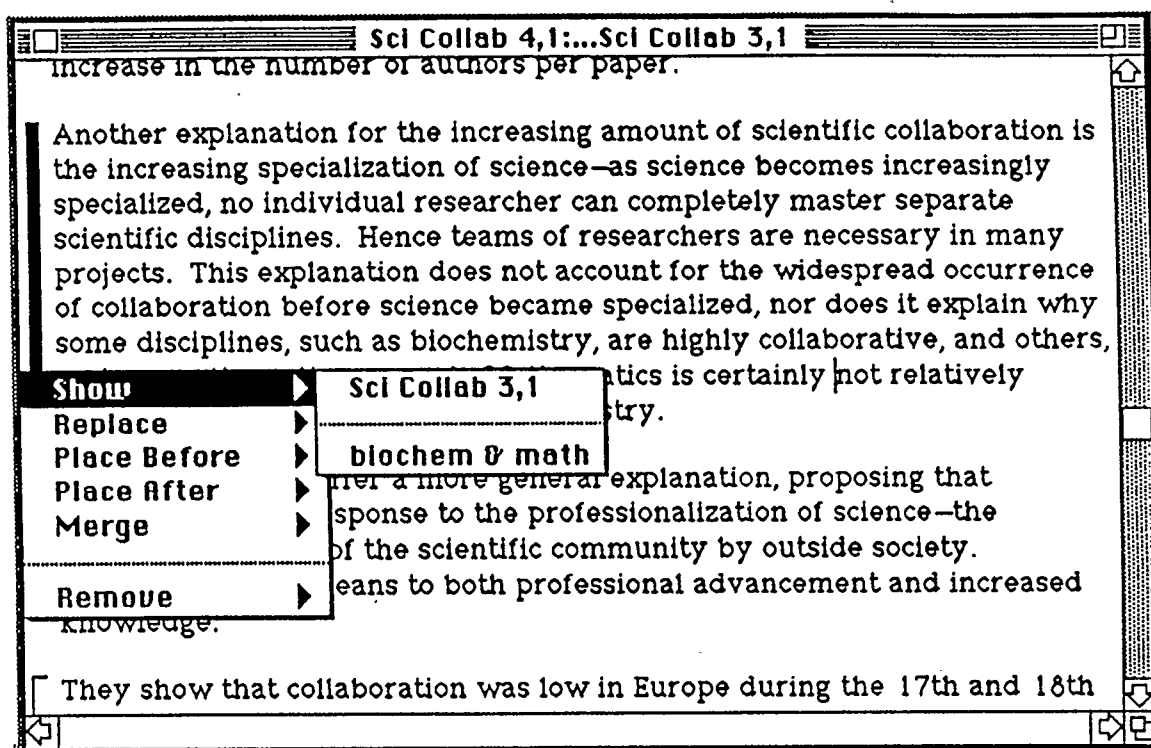


Figure 5.7: The Popup Menu with an Annotation

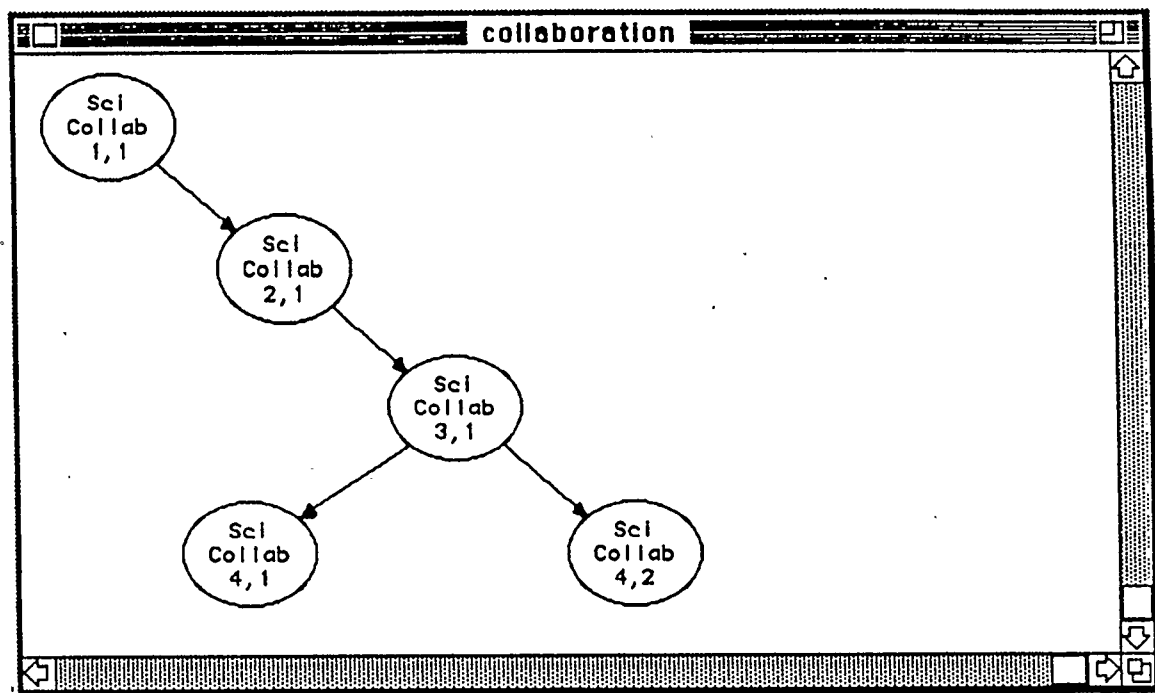


Figure 5.8: The Document Version Browser



By holding down the shift key when clicking on a document version, more than one version can be selected. Double clicking on a document version opens an edit window containing that version, with any other selected versions being searched for alternative paragraphs.

### 5.2.5 Import/Export of Document Versions

To encourage the widespread use of GroupWriter, compatibility with existing word processors is important. GroupWriter can import files or export document versions in one of three formats: plain text, styled text, and Rich Text Format (RTF). Plain text consists of a string of characters. Styled text contains additional formatting information, and is used by the TextEdit facilities in the Macintosh Toolbox. RTF has much more powerful formatting capabilities than styled text, and is recognized by WriteNow and Microsoft Word on the Macintosh. In addition, RTF is the clipboard format used for Microsoft Windows 2.0 and 3.0, and is recognized by Microsoft Word for MS-DOS.

A document version is exported by selecting **Export** from the **File** menu. A dialog box appears, allowing the user to specify a name and directory for the file (Figure 5.9). Clicking on the **Format** button allows the file format to be chosen (Figure 5.10).

Importing a file is done in the same way as opening a structure file, by selecting **Open** from the **File** menu. GroupWriter can distinguish between four types of files: structure files, plain text files, styled text files, and RTF files. When a file is imported into GroupWriter, it is treated as being the initial version of a new document.

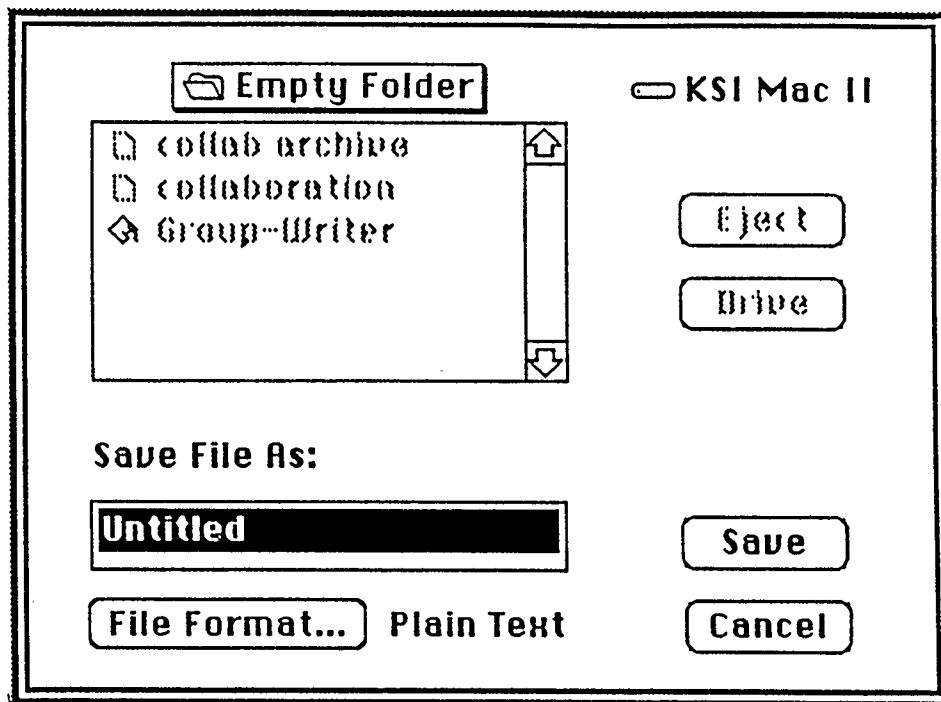


Figure 5.9: Exporting a Document Version

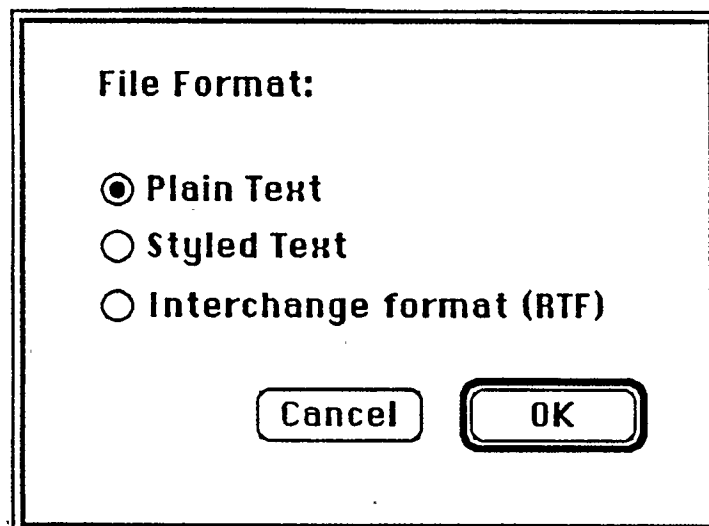


Figure 5.10: The File Format Dialog

## Chapter 6

### Implementation of GroupWriter

GroupWriter was implemented using THINK C [BM89] on a Macintosh SE/30 computer. THINK C provides object oriented extensions to the programming language C, implementing a subset of C++. This chapter discusses the major architectural features of GroupWriter.

#### 6.1 Overall Architecture

The architecture of GroupWriter is shown in Figure 6.1. Arrows show dependencies; for example, the text editor depends on the monitor to receive keyboard and mouse input. The design of GroupWriter is modular, and as far as possible there are only minimal dependencies between the different components of the system. This allows any of the components to be experimented upon and changed without a major rewrite of code in other components.

#### 6.2 User Interface Design

GroupWriter is currently implemented for the Apple Macintosh series of computers, and in general its user interface conforms to the Apple Human Interface Guidelines [App87]. Other user interface guidelines [Nie90] were also consulted.

Given the complexity of managing both historical and alternative versions of a document, the user interface to a group-writing tool is extremely important. Group-

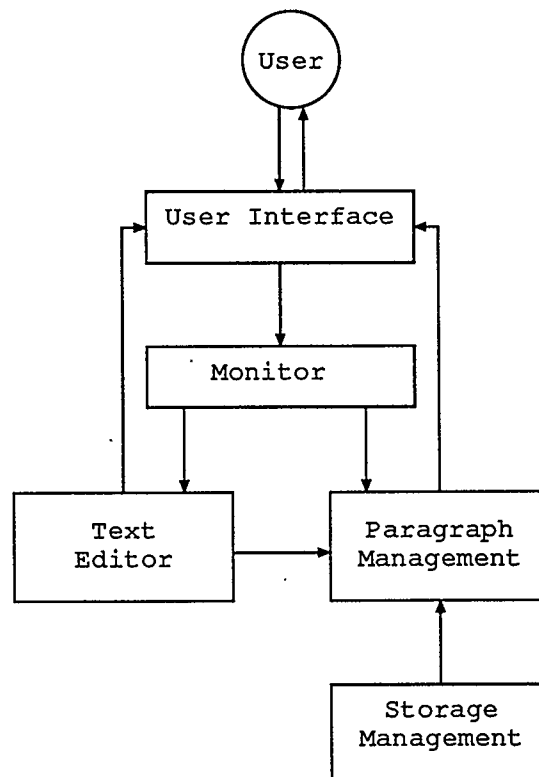


Figure 6.1: The Architecture of GroupWriter

Writer has several critical features that must be clearly presented to users:

- The existence of annotated paragraphs.
- The existence of alternative versions of paragraphs.
- The existence of, and relationships between, different versions of a document.

To facilitate the widespread acceptance of GroupWriter, these features are designed to be as unobtrusive as possible. GroupWriter is intentionally similar in operation to existing Macintosh word processors such as WriteNow and Word. Users without a sophisticated awareness of the underlying archiving and versioning process can select document versions and edit them in a conventional word-processing window. This allows a group to structure itself such that some writers can create and edit documents conventionally while others take responsibility for reconciling versions.

Support for version management is also structured such that users need not use facilities widely different from those in a conventional word processor. Users can track versions through naming conventions and open alternative document versions through a standard file opening dialog. In addition, a graphical interface showing the relationships between document versions is also available, as discussed in Chapter 5.

The presence of paragraph alternatives and of paragraph annotations is shown by a bracket in the left-hand margin. These brackets are modelled on those used by Mathematica [Wol89], which uses nested brackets in representing different components of mathematical equations. Brackets were chosen because they are unobtrusive. In the future, if GroupWriter incorporates structured editing or outlining capabil-

ities, nested brackets could be used to represent alternative versions of sections of a document.

Another feature of GroupWriter's user interface deserving of attention is the merging of paragraphs. Paragraphs are compared on a sentence-by-sentence basis using a variant of the Unix *diff* utility [HM76], which is normally used to compare files on a line-by-line basis. Sentences were chosen as the unit of comparison because they are a more natural conceptual unit for writers.

When merging, differences between paragraphs are displayed in a separate window. This window is movable; it may be positioned so that the contents of any window beneath it are unobscured. It is also a *floating window*, meaning that no other window can be placed in front of it. These factors encourage users to complete the merge before carrying out any further editing operations in the original edit window.

## 6.3 Detailed Architectural Description

This section covers in greater detail the main architectural components of GroupWriter and the design decisions made.

### 6.3.1 Storage Management

GroupWriter uses two files to store document versions. The *archive file* stores the text of paragraphs and annotations occurring in any of the versions. The *structure file* records which paragraphs and annotations are contained in each of the document versions. This organization was chosen for ease of implementation.

The only assumption made about information stored in the archive file is that it must be immutable. Everything stored in the archive file is regarded as a variable sized contiguous block of bytes, thus a paragraph is stored as a variable sized block of characters. GroupWriter maintains all versions of a document, so nothing stored in the archive file is ever deleted. Hence there are no concerns about internal fragmentation, and any new information can simply be appended to the file.

The structure file contains all information necessary to reconstruct any of the document versions from the paragraphs and annotations stored in the archive file. Because document versions are immutable and are never deleted, information about each new version can be appended to the structure file, speeding up the Save operation. Figure 6.2 shows the information in the structure file.

The document version record is a C structure with the following format:

```
typedef struct {  
    Str31  name;           /* name of document version */  
    short  id;             /* id number of version */  
    short  parent;         /* id of parent version */  
    short  level;          /* level in version tree */  
    short  numParas;       /* number of paragraphs */  
} versionRec;
```

The text of paragraphs and annotations is stored in the archive file. In the structure file, a record of all the paragraphs and annotations occurring in each document version is retained. To record the location of a paragraph or an annotation in the archive file, the following structure is used.

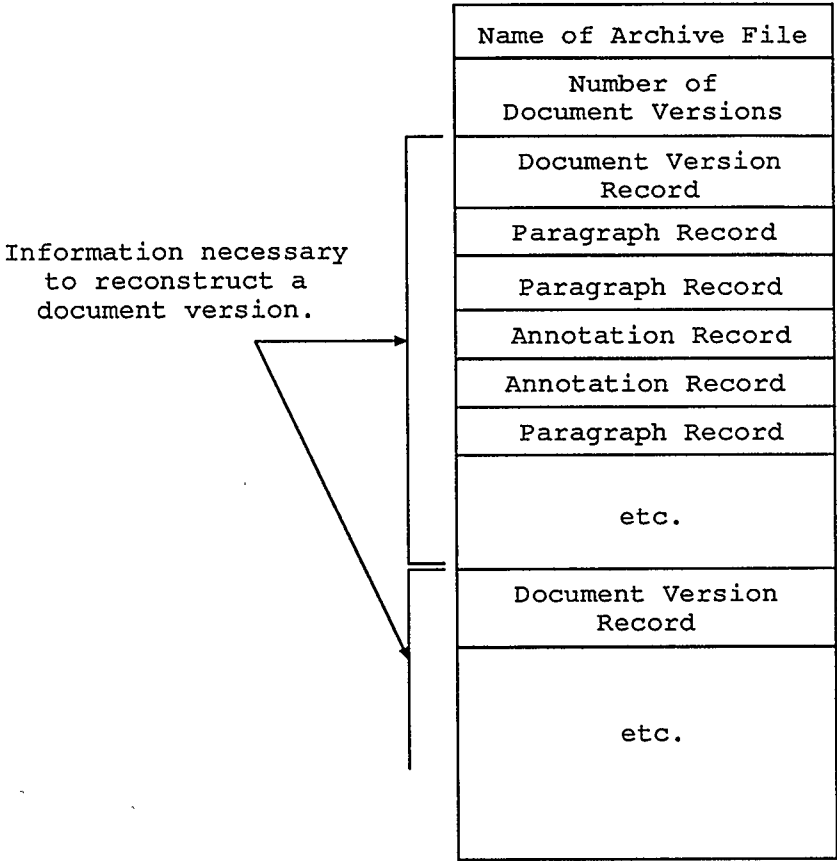


Figure 6.2: Storing a Document Version



```
typedef struct {  
    long  textAddr;    /* address of text */  
    long  textLen;     /* length of text */  
    long  styleAddr;   /* address of style */  
    long  styleLen;    /* length of style */  
} paraLocnR;
```

The `textAddr` and the `textLen` fields give the address (offset) and the length of the paragraph text in the archive file. The `styleAddr` and the `styleLen` fields give the address and the length of the corresponding formatting information (font, font size, etc.).

For each paragraph in a document version, a record giving its location in the archive file, the ID number of the original version of the paragraph, and the number of attached annotations is maintained:

```
typedef struct {  
    paraLocnR  location; /* location in archive file */  
    long        ancestor; /* identity of original version */  
    short       numAnn;    /* number of annotations */  
} paraInfoRec;
```

The identity of a paragraph's ancestor is given by the address of the ancestor in the archive file.

Annotation records follow immediately after the paragraph to which they are attached, and have the following structure.

```
typedef struct {
```

```

    Str31      ptitle;          /* annotation title */
    paraLocnR  location;        /* location in archive file */
} annInfoRec;

```

Typically the structure and the archive files will be located on an AppleShare file server. Care needs to be taken when several users attempt to access simultaneously these files. Simultaneous access is permitted if users only wish to read the files. Users wishing to change the files, however, must be granted exclusive access to *both* of the files at the *same time*, in order to keep the information in both files complete and consistent: file updates must be atomic actions [JC85], in order to appear as single, indivisible operations to other users.

### 6.3.2 Paragraph Management

During editing, GroupWriter needs to keep track of the location and the identity of each paragraph. Paragraph locations are needed to draw correctly the brackets which indicate the presence of alternative paragraphs or of annotations. Paragraph identities are needed to determine which paragraphs the effects of a Merge operation should be applied to.

The following data structure is maintained for each paragraph:

```

typedef struct paraR {
    long      beginPtr;        /* paragraph beginning */
    long      id;              /* paragraph ID */
    Boolean    dirty;          /* paragraph modified? */
    short      docVersion;     /* document version */
}

```

```

    paraLocnR    location;        /* location in archive */
    long         ancestor;        /* address of original para */
    Handle       theTextData;     /* paragraph text */
    Handle       theStyleData;    /* paragraph format */
    struct paraR **altParas;      /* alternative paras */
    annRec       **annots;        /* annotation list */
} paraRec;

```

The offset of the paragraph from the beginning of the text is given by `beginPtr`. `id` is the unique ID of the paragraph, assigned when the paragraph record is created. `dirty` is true if the paragraph has been modified since editing began, otherwise false. `docVersion` identifies the document version containing this paragraph. When a document version is opened, an array is constructed containing the names of all the document versions. The `docVersion` field is an index into this array. `location` contains the location of the paragraph in the archive file, or `-1` if the paragraph is new. `theTextData` and `theStyleData` are `NULL` if the paragraph is currently in the edit window, otherwise (the paragraph is in the alternative list) they contain the paragraph text and formatting information.<sup>1</sup> `altParas` is a list containing the alternative versions of the paragraph, and `annots` is a list containing the annotations of the paragraph.

Each annotation is represented by the following data structure:

```

typedef struct annR {
    paraLocnR    location;        /* location in archive */

```

---

<sup>1</sup>Both these fields are of type `Handle`. A `Handle` is a Macintosh Memory Manager data type, and may be loosely regarded as a pointer to a block of memory.

```

    Str31      ptitle;          /* annotation title */
    Handle     theTextData;     /* annotation text */
    Handle     theStyleData;    /* annotation format */
    struct annR **nextAnn;      /* next annot in list */
} annRec;

```

When opening a document version, the secondary versions are examined to find any alternative versions of paragraphs in the primary version. The pseudo-code to perform this operation is given below.

```

for every secondary version
  for every paragraph in the primary version {
    primID = ID of primary paragraph;
    primAncestor = ID of primary paragraph's ancestor;
    for every paragraph in the secondary version {
      secID = ID of secondary paragraph;
      secAncestor = ID of secondary paragraph's ancestor;
      if (primAncestor == secAncestor and primID != secID)

        have found an alternative version of

                                the primary paragraph

    }
  }

```

If there are  $s$  secondary versions, and an average of  $n$  paragraphs in each document

version, then this is an order  $sn^2$  operation. Although  $s$  would typically be small in comparison with  $n$ , this is still an expensive operation. It is only performed once, however, at the beginning of an editing session.

## Chapter 7

### Evaluation of GroupWriter

The chapter evaluates some aspects of the performance of GroupWriter. First, the storage efficiency of GroupWriter is compared with that of RCS. Second, the results of some preliminary user trials with GroupWriter are presented.

#### 7.1 Comparison with RCS

This section compares the storage efficiency of GroupWriter with that of RCS. Comparisons with other systems such as Quilt or PREP would also be useful, but unfortunately those systems were unavailable.

RCS [Tic82] stores a complete copy of only the most recent version of a document, and uses negative deltas to reconstruct earlier versions.<sup>1</sup> In calculating the differences between two document versions, RCS uses the Unix diff utility, to compare the versions on a line-by-line basis.

Interestingly, diff considers a line to be a sequence of characters terminated by a <return>, the same definition used for a GroupWriter paragraph; a diff line is equivalent to a GroupWriter paragraph. RCS stores the lines which differ from one version to another, and GroupWriter the paragraphs, hence they both store the same thing.

This is an unfair comparison however. RCS was developed to run on Unix

---

<sup>1</sup>The situation is more complex when the version history of an RCS document contains branches, for details see [Tic82].

Draft of Document	RCS	GroupWriter			
		Archive text	style	Structure	Total
Initial	30991	30822	2904	3085	36811
Revision 1	32701	34089	3102	5783	42974
Revision 2	34481	37942	3256	8481	49679
Revision 3	36190	42806	3432	11179	57417
Revision 4	37958	45549	3564	13877	62990

Table 7.1: Comparison of GroupWriter and RCS: Case I. All measurements are given in bytes.

systems, where a line is typically not the same as a paragraph. Many editors available on the Unix system do not wrap words at the right margin as do WYSIWYG editors on the Macintosh such as GroupWriter. GNU Emacs and vi are two Unix editors that can wrap words at the right margin, however they insert a <return> character in doing so. This means that a paragraph is not equivalent to a line when using these editors. The files that RCS is commonly used to store do not have a paragraph-line equivalence.

To compare the storage requirements of GroupWriter with those of RCS, a sample paper 4227 words in length was used. The paper contained 587 lines and 132 paragraphs. In the comparison, blocks of 5 lines were changed by adding one character in moving from one document version to another. Two cases were considered, and in both the paper was revised four times. In the first case, 5 randomly chosen blocks were changed in moving from one version to another. In the second case, 10 randomly chosen blocks were changed in moving from one version to another. The results of the first case are in Table 7.1, the second in Table 7.2.

In both cases, a linear increase in storage space is required in moving from

Draft of Document	RCS	GroupWriter			
		Archive text	style	Structure	Total
Initial	30991	30822	2904	3085	36811
Revision 1	34113	36365	3168	5783	45316
Revision 2	37275	43518	3454	8481	55453
Revision 3	40109	51092	3718	11179	65989
Revision 4	43147	58419	4136	13877	76432

Table 7.2: Comparison of GroupWriter and RCS: Case II. All measurements are given in bytes.

one version to another, for both GroupWriter and RCS. Nonetheless, GroupWriter requires significantly more space, especially as more revisions are made.

This difference is partially explained by the additional style information (font, font size, etc.) which is stored by GroupWriter and not by RCS. GroupWriter's structure file also requires additional space. Comparing the amount of text stored in the archive file with the size of the RCS file shows that, for the examples tested, storing the lines which differ between versions is more efficient in terms of storage utilization.

## 7.2 Preliminary User Trials

Two approaches were used during the preliminary user trials for GroupWriter. First, users were given a brief demonstration of GroupWriter's main features, allowed to experiment with the system for a while, and asked to complete a questionnaire. Second, GroupWriter was used to produce a document in a group situation.



### 7.2.1 Questionnaire Results

As part of the user trials for GroupWriter, a survey was conducted in which subjects were asked to complete a questionnaire after being given a brief demonstration of GroupWriter's main features and being allowed to experiment with the system for a while. The goals of the survey were twofold:

1. To find out which aspects of group-writing people felt were important in a group-writing tool.
2. To collect general reactions to GroupWriter, and suggestions for improvement.

Appendix A contains one of the completed questionnaires obtained during the survey. In designing the questionnaire, reference was made to the QUIS questionnaire [Lab87] developed at the Human Computer Interaction Laboratory at the University of Maryland.

Table 7.3, derived from Part I of the questionnaire, shows the personal details of the subjects. Four of the subjects were female, three male. As GroupWriter is ultimately intended for use by a wide range of people, an attempt was made to survey people with a non computer science background. Only two of the subjects had a computer science background—the two students.

Table 7.4, derived from Part II of the questionnaire, shows previous Macintosh experience of the subjects. Most of the subjects were experienced Macintosh users, and all had prior experience with Microsoft Word, a single-user word processor for the Macintosh.

Table 7.5, derived from Part III of the questionnaire, shows the importance attached by the subjects to various factors related to group-writing. These factors

User	Sex	Occupation	Age
1	F	Admin. Assistant	35-45
2	F	Biochemist	25-35
3	M	Professor	35-45
4	M	Student	under 25
5	M	Student	25-35
6	F	Technical Communicator	25-35
7	F	Admin. Assistant	not given

Table 7.3: User Profiles

User	Length of time have used Macintosh	Number of frequently used Macintosh applications	Applications
1	> 3 years	4	Word, Excel, Bedford Simply Accounting
2	2-3 years	1	Word
3	> 3 years	3	Word, Excel, MacDraw
4	1 wk to 1 mth	1	Word
5	> 3 years	> 6	Word, SuperPaint, THINK C, ZTerm, ResEdit, MacDraw II, miscellaneous games
6	> 3 years	4	Word, MacDraw, SuperPaint, CricketGraph
7	> 3 years	2	Word, Excel

Table 7.4: Previous Experience

User	History Versions	Parallel Versions	Version Comparison	Annotation	Compatibility	Simple, Natural	Electronic Mail	Privacy
1	8	9	10	8	4	10	—	1
2	10	10	10	10	7	10	8	10
3	10	8	10	10	8	6	8	1
4	5	10	5	10	10	1	—	10
5	10	8	10	4	9	9	7	—
6	8	7	10	8	6	6	4	4
7	9	9	9	9	9	10	6	10

Table 7.5: Importance of Various Factors to Group-Writing (on a scale of 1 to 10)

correspond to many of the requirements derived in Chapter 2 for a group-writing system. The subjects rated the importance of each factor on a scale of 1 (not important) to 10 (very important). It was expected that all these factors would be considered important by the subjects. In general, the subjects felt that having history versions of a document, parallel versions, version comparison, and annotation are important. Of lesser importance are compatibility with other editors, similarity to existing Macintosh editors, and support of electronic mail. The subjects' responses to the importance of ensuring privacy varied widely. Most of them considered it unimportant, but one of the subjects, a biochemist, considered privacy to be extremely important and remarked that "plagiarism and pirating of data is very rampant".

Finally, Part IV of the questionnaire, which was intended to collect general comments about group-writing and GroupWriter, is discussed. All of the subjects reacted positively to the features offered by GroupWriter, and all said that they would

User	Number of Coauthors	Regularly used Features				
		Finding alt. paragraphs	Merging alt. paragraphs	Paragraph annotation	Graphical Browser	RTF Import/Export
1	up to 15	y	y	y		
2	3 or more	y	y	y	y	
3	1 or 2	y	y	y	y	
4	2		y	y	y	
5	2	y	y		y	
6	none	y		y		y
7		y	y	y	y	y

Table 7.6: General Responses to the Questionnaire

like to use a group-writing system in their work. Most of the subjects reported typically having two coauthors when writing a paper, though the administrative assistant reported having up to 15 (Table 7.6). The technical communicator typically worked as the sole writer in a group, with other group members annotating the document versions.

In general, the subjects said the most important features in a word processor supporting group-writing were annotation, comparison of document versions, and facilities to support the merger of document versions. These results imply that maintaining different versions of a document is important.

Several problems with GroupWriter emerged during the user trials. The most common complaints involved the Merge operation, which was generally felt to be too limited. The subjects suggested that Merge should allow editing in the merge window, more control over where text would be placed in the merged paragraph, and the merger of more than two paragraphs at one time. Another complaint was that the algorithm used by Graph to layout the version tree is not very good. Some problems with the user interface were also revealed. Some of the subjects would inadvertently select Replace instead of Show on the popup menu 5.4, because in selecting the paragraph to show the mouse pointer tends to move to the right and down at the same time. Additionally, after typing the title of an annotation 5.6 some users tended to type <return>, which performs the same operation as clicking in the OK button. This could be fixed by disabling <return>.

In general, the subjects indicated that if they used GroupWriter, the features they would use regularly would be finding alternatives of paragraphs, merging alternative paragraphs, annotation of paragraphs, and the graphical browser. RTF import/export capability was considered less important.

Finally, a word of caution regarding the results of the survey. None of the subjects had regularly used a computer-based group-writing system in their work, and for most of them GroupWriter was the first group-writing system they had seen. Hence it is likely that their answers in the questionnaire were largely given with respect to GroupWriter.

### 7.2.2 Trial Use of GroupWriter

As part of its evaluation, GroupWriter was used collaboratively to produce a sample document. In particular, this author and another user collaborated on the production of a sample column describing GroupWriter for the Calgary Herald newspaper. The author's comments and observations will not be reported here because of his extreme familiarity with GroupWriter.

GroupWriter was found to be useful in collaboratively producing the document. In particular, annotation provided a useful way to communicate concerns and opinions to the other author, and versioning allowed changes from one version to another to be easily isolated.

Some problems with GroupWriter also emerged. First, at present annotations only apply at a paragraph level. This caused problems when an annotation was intended to apply to the entire document, or to a single word or sentence within a paragraph. Second, the paragraph bracket for alternative paragraphs and for annotations is the same. This should be changed, so that the bracket clearly indicates whether an annotation is present, or an alternative paragraph. Third, the creator of each annotation is not recorded and can be difficult to remember.

Furthermore, it was found that during the initial stages of the project changes between document versions were relatively large. As the document neared completion, however, changes were restricted to only a word or phrase within a paragraph. The Merge operation allows paragraphs to be compared on a sentence-by-sentence basis. This worked well during the initial phases of the project. At the end of the project, when differences between paragraphs were often only a single word, Merge

did not work so well. Though it indicated which sentences differed, the differences were hardly noticeable.

## Chapter 8

### Conclusions and Future Work

The future directions in which the work reported in this thesis could be continued are discussed in this chapter. A brief summary of achievements is then presented.

#### 8.1 Future Work

There are many ways in which GroupWriter could be extended, and they are outlined below.

##### 8.1.1 Apple System 7.0

Apple will soon release System 7.0 [App90], the new version of the Macintosh Toolbox and Operating System. System 7.0 contains several components that could be incorporated into future versions of GroupWriter: the Alias Manager, the Edition Manager, the Event Manager, and the Help Manager.

The Alias Manager can assign a unique identification number—the file ID—to a file. The Alias Manager can use the file ID to locate the file, even if it is moved to another folder or to another volume. Recall that currently an archive file must reside in the same folder as the corresponding structure file; if the archive file is moved to another folder or volume, GroupWriter will be unable to find it again. The Alias Manager will allow the archive and the structure files to be stored in different folders.

The Edition Manager facilitates the sharing of data between documents. Within



a document users can define *sections* which are to be incorporated into other documents. Whenever a section in the original document is changed, the other documents can be automatically updated. For example, a user could define a MacDraw picture to be a section, and include this section in a Microsoft Word document. Whenever the original MacDraw picture is changed, the changes can be automatically incorporated into the Microsoft Word document. GroupWriter could be extended to support the features of the Edition Manager, thus allowing it to share data with other applications that support the Edition Manager.

Applications using the Event Manager can send high-level events to each other, which allows applications to share functionality. While working in one application, commands and capabilities of other applications—even from different software vendors—can be used. For example, GroupWriter could incorporate elements of a drawing application.

The Help Manager could be used to display help messages for GroupWriter's menus, windows, and dialogs. When help is enabled, the Help Manager can provide help in the form of text or a picture enclosed in a help balloon (Figure 8.1) for the part of the display under the mouse pointer. Normal operation of an application can continue at the same time that help is enabled.

### 8.1.2 Document Storage

GroupWriter's current document storage scheme has the advantage of being simple. Unfortunately, it suffers from two major problems. It requires too much storage space, and reconstructing large document versions is too slow.

Data compression could be used to reduce storage requirements, but would slow

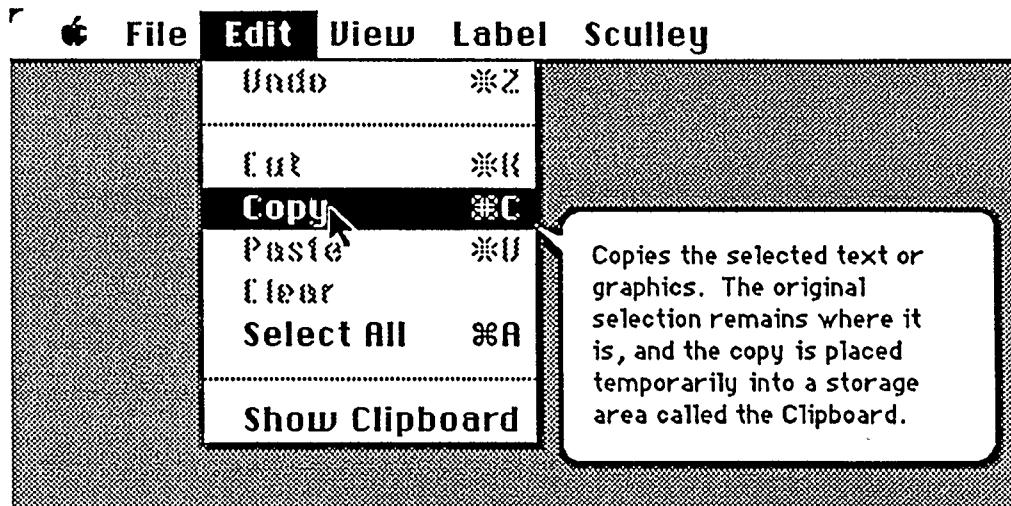


Figure 8.1: The Help Manager

down document reconstruction even further. Another possibility is to calculate differences between document versions at a lower level of granularity—at the sentence, line, or word level. The possibility of deleting versions of a document could be examined. This is not a trivial issue; what happens to the link between a document version and its parent if the parent is deleted?

To improve the speed at which document versions are reconstructed, the RCS approach could be adopted, meaning that the most recent version of a document would be stored in its entirety. Other document versions would be reconstructed relative to this version.

### 8.1.3 Improvement of Existing Features

#### Word Processing Functionality

GroupWriter currently derives its word processing functionality from the TextEdit facilities in the Macintosh Toolbox. TextEdit allows different fonts and styles to be

used within the same document, and provides basic editing operations such as Cut and Paste. Its editing capabilities are limited, however, because it was originally designed for use in dialogs. In particular, it cannot operate on documents with more than 32767 characters, and its performance degrades well before this limit.

Datapak's Word Solution Engine is a powerful word processor which can be invoked from a program similarly to Macintosh Toolbox routines. It provides much more word processing functionality than TextEdit, and can be upgraded to incorporate memory management facilities to handle large documents. It is planned that future versions of GroupWriter will use Word Solution Engine instead of TextEdit.

Another limitation of GroupWriter's current editing facilities is its inability to handle figures. Word Solution Engine can easily be modified to handle figures. Further, it would be easy to isolate alternative versions of figures from one document version to another, as can currently be done for paragraphs.

Even with the addition of Word Solution Engine, GroupWriter will not have any outlining or structured editing capabilities. One possibility is to allow paragraphs to be grouped together hierarchically. Annotations could then apply to sections of a document, and the popup menu could apply to sections, or groups of paragraphs, in a document. If structured editing capabilities are added to GroupWriter, then the option of allowing a document version to be saved in SGML format should be considered.

### **Annotation**

Annotation as it currently exists in GroupWriter can be extended in many ways. First, annotation of annotations can be allowed. This would allow annotations to be

nested to an arbitrary depth. Second, the scope of annotations could be expanded to allow the inclusion of graphics and sound. These capabilities could also be added to the text of document versions. Third, annotations and alternative paragraphs could optionally be displayed in a grid or matrix as in the PREP system [NKCM90], instead of in a list contained within a popup menu. Fourth, if outlining or structured editing facilities are added to GroupWriter, then both annotations and the brackets in the left margin of the edit window could apply to sections of a document version as well as to individual paragraphs. Fifth, the ability to annotate individual words or sentences could be added.

### **The Popup Menu**

The popup menu needs to be changed in several ways. First, the user survey indicated that some users tend to accidentally select `Replace` instead of `Show`. This can be solved by separating `Show` from the other selections in the menu, similarly to `Remove`. Second, if sound annotation is added to GroupWriter, the popup menu may have to be redesigned.

### **Extension of the Merge Operation**

The paragraph merge operation, which merges alternative versions of a paragraph, could be extended to operate in a similar fashion on two entire document versions. Whether versions should be compared on a sentence-by-sentence basis, or on a paragraph-by-paragraph basis, is open to further investigation. Another possibility is to support both of these options, allowing the user to specify which they prefer as the default. If outlining or structured editing facilities are added to GroupWriter, then the merge operation could also operate on a section-by-section basis.

As was uncovered during the user trials, the functionality of the Merge operation could be extended in several ways. Editing within the Merge window, and the merger of more than two paragraphs at one time, could both be supported. The text that differs between paragraphs could be highlighted. Finally, more control over where merged sentences are placed in the final paragraph could be offered.

### **The Browser**

The browser used for graphically displaying the versions of a document could be changed to use a better tree drawing algorithm [BKW89]. It could also be enhanced to optionally allow the creator and the creation date of each version to be displayed. Further, document versions could be annotated.

### **Window Management**

The window management capabilities of GroupWriter could be much enhanced. Currently, if one window is totally obscured by another, the only way to bring it to the front is by closing or resizing the obscuring window. A more elegant approach is to use a Windows menu as in Microsoft Word. This menu contains a list of all windows, and the user can select which they wish to bring to the front of the screen. This menu could also have commands for zooming/unzooming windows, for closing or saving all windows, and for moving the foremost window behind all other windows.

Another possibility is to allow windows to be *grouped* together. Operations such as scrolling or closing a grouped window are automatically applied to all other windows in the group. For example, two windows could be synchronously scrolled to compare two document versions on a line-by-line basis.

How can windows be grouped together? Group and UnGroup options can be added

to the **Windows** menu. When **Group** is selected, any windows can be selected to be part of the group. A double click ends the grouping process. The **UnGroup** command is similar, and is used to remove windows from a group.

#### 8.1.4 Additional Features

##### User Identity

As shown by the user trials, it is currently difficult to remember which author created which annotation or document version. This problem becomes particularly acute with larger groups. Hence it is important that GroupWriter record the name of each author and the date that changes are made.

##### Security

No special security features are implemented for GroupWriter. Existing Macintosh word processors could be used to read paragraphs from the archive file, and users with a knowledge of the format of the structure file could easily reconstruct document versions. The best way to keep a document containing sensitive information secure is currently to store it on a floppy disk or in a password protected folder on an AppleShare server. Further work is needed to see how additional security could be implemented, and whether it would be valued by potential users.

Indeed, enhancing GroupWriter so that sensitive information is guarded from unauthorized access is extremely difficult. The root of the problem lies in the lack of usernames and passwords on the Macintosh Multifinder, unlike other operating systems such as Unix or VMS. GroupWriter could request usernames and passwords from users each time a document is to be edited or viewed, however, this would likely

be an imposition resented by users unfamiliar with such conventions.

### **Electronic Mail**

Currently, GroupWriter's only feature that explicitly supports communication between collaborating writers is annotation of paragraphs. This could be augmented in future by the addition of electronic mail. As with Quilt, electronic mail could be used to notify authors when substantial changes are made to a document version, or when the deadline is nearing for completion of the document.

Electronic mail could be added in two ways. First, GroupWriter could have its own electronic mail system, with messages presumably being stored either in the document archive or in another special file. Second, GroupWriter could be adapted to use existing electronic mail systems for the Macintosh, such as Microsoft Mail.

#### **8.1.5 Document Version Comparison**

GroupWriter currently compares document versions on a paragraph-by-paragraph basis. However, the comparison does not use any knowledge of the relative position of each paragraph in the text of each document versions. Hence the comparison will not reveal if the relative order of paragraphs has changed from one document version to another. It will also not reveal if one version contains paragraphs that another does not. The comparison algorithm used by GroupWriter in future must be changed to account for these two issues, though the issue is not trivial because more than two document versions may be compared at the same time.

### 8.1.6 User Trials

Much more extensive user trials should be conducted for GroupWriter. There are many interesting questions to address. How can the user interface of GroupWriter be improved? What additional features need to be supported? How is GroupWriter used by collaborating authors? How does it change the way that people write or work? It is probably important to upgrade the word processing functionality of GroupWriter closer to that in commercial word processors before detailed human factors studies are undertaken.

## 8.2 Conclusions

This thesis has described the design and the implementation of a word processor for group writing. The need for such a system was established by an examination of group-writing trends in the scientific and the business communities. Through an examination of group writing in the scientific community, and of the writing process itself, a general set of requirements to be satisfied by a group-writing system were derived.

Table 8.1 contains a comparison of GroupWriter with other group-writing systems. The novel features that distinguish GroupWriter from other similar systems are listed below:

1. The ability to isolate easily alternative versions of a paragraph.
2. The use of paragraph bracketing and a popup menu to display alternative paragraphs and to perform operations on them.



	Hypertext Systems				Linear Systems		
	Group-Writer	Contexts	Quilt	PREP	CES	Shared Books	RCS
History versions	y	y	y	y	y	?	y
Parallel versions	y	y	?	y	n	n	y
Version comparison	y	y	y	n	n	n	y
Annotation	y	y	y	y	y	y	n
Compatible with other editors	y	?	?	y	?	?	y
Some users need not manage versioning	y	n	y	y	y	y	y
Electronic mail	n	?	y	n	n	n	n

Table 8.1: A Comparison of Group-Writing Systems

3. The ability to compare and to merge paragraphs using a sentence-by-sentence comparison.

Finally, some preliminary user trials with GroupWriter were conducted. These trials not only indicated that GroupWriter would be useful in a group-writing situation, but were also extremely helpful in isolating those areas of GroupWriter that need improvement.

## Bibliography

- [AFQ89] Jacques Andre, Richard Furuta, and Vincent Quint. By way of an introduction. Structured documents: What and why? In J. Andre, R. Furuta, and V. Quint, editors, *Structured Documents*, pages 1–6. Cambridge University Press, 1989.
- [App87] Apple Computer, Inc. *Human Interface Guidelines: the Apple Desktop Interface*, 1987.
- [App90] Apple Computer, Inc. *Inside Macintosh, Volume VI*, May 1990. Preliminary Draft.
- [BBJB87] Lillian Bridwell-Bowles, Parker Johnson, and Steven Brehe. Composing and computers: Case studies of experienced writers. In Ann Matsuhashi, editor, *Writing in Real Time: Modeling Production Processes*, pages 81–107. Ablex Publishing Corporation, 1987.
- [Beg90] John Begoray. The development of computer based software tools to support the organizing process of expository writing: Some theoretical issues and implications. Master's thesis, The University of Calgary, April 1990. Department of Educational Psychology.
- [BKW89] A. Bruggemann-Klein and D. Wood. Drawing trees nicely with T<sub>E</sub>X. *Electronic Publishing*, 2(2):101–115, July 1989.
- [BM89] Philip Borenstein and Jeff Mattson. *THINK C: User's Manual*. Symantec Corporation, 1989.
- [BMNS87] G. S. Blair, J. A. Mariani, J. R. Nicol, and D. Shepherd. A knowledge-based operating system. *The Computer Journal*, 30(3):193–200, June 1987.
- [Bry88] Martin Bryan. *SGML: An Author's Guide to the Standard Generalized Markup Language*. Addison-Wesley, 1988.
- [Bus45] Vannevar Bush. As we may think. *The Atlantic Monthly*, July 1945.
- [CBY89] Timothy Catlin, Paulette Bush, and Nicole Yankelovich. InterNote: Extending a hypermedia framework to support annotative collaboration. In *Hypertext'89 Proceedings*, pages 365–378, Pittsburgh, Pennsylvania, November 1989.

- [CD89] Christodoulos Chamzas and Donald L. Duttweiler. Encoding facsimile images for packet-switched networks. *IEEE Journal on Selected Areas in Communications*, 7(5):857–864, June 1989.
- [CIV86] Giovanni Coray, Rolf Ingold, and Christine Vanoirbeek. Formatting structured documents: Batch versus interactive? In J. C. van Vliet, editor, *Text Processing and Document Manipulation*, pages 154–170. Cambridge University Press, April 1986.
- [Con87] Jeff Conklin. Hypertext: An introduction and survey. *IEEE Computer*, 20(9):17–41, September 1987.
- [CRD87] James H. Coombs, Allen H. Renear, and Steven J. DeRose. Markup systems and the future of scholarly text processing. *Communications of the ACM*, 30(11):933–947, November 1987.
- [Cro90] Geoffrey A. Cross. A Bakhtinian exploration of factors affecting the collaborative writing of an executive letter of an annual report. *Research in the Teaching of English*, 24(2):173–203, May 1990.
- [CW84] J. Cleary and I. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, COM-32(4):396–402, April 1984.
- [Dai86] Colette Daiute. Physical and cognitive factors in revising: Insights from studies with computers. *Research in the Teaching of English*, 20(2):141–159, May 1986.
- [Dal87] Richard Dalton. Group-writing tools: Four that connect. *Information Week*, pages 62–65, Mar. 9 1987.
- [dBR78] D. deB. Beaver and R. Rosen. Studies in scientific collaboration part I. The professional origins of scientific co-authorship. *Scientometrics*, 1(1):65–84, 1978.
- [dBR79a] D. deB. Beaver and R. Rosen. Studies in scientific collaboration part II. Scientific co-authorship, research productivity and visibility in the French scientific elite, 1799–1830. *Scientometrics*, 1(2):133–149, 1979.
- [dBR79b] D. deB. Beaver and R. Rosen. Studies in scientific collaboration part III. Professionalization and the natural history of modern scientific co-authorship. *Scientometrics*, 1(3):231–245, 1979.

- [Dic86] David K. Dickinson. Cooperation, coolaboration, and a computer: Integrating a computer into a first-second grade writing program. *Research in the Teaching of English*, 20(4):357–378, December 1986.
- [DS87] Norman M. Delisle and Mayer D. Schwartz. Contexts—a partitioning concept for hypertext. *ACM Transactions on Office Information Systems*, 5(2):168–186, April 1987.
- [DS89] Norman M. Delisle and Mayer D. Schwartz. Collaborative writing with hypertext. *IEEE Transactions on Professional Communication*, 32(3):183–188, September 1989.
- [EGR91] Clarence Ellis, Simon Gibbs, and Gail Rein. Groupware: Some issues and experiences. *Communications of the ACM*, 34(1):38–58, January 1991.
- [FH81] Linda S. Flower and John R. Hayes. A cognitive process theory of writing. *College Composition and Communication*, 32:365–387, 1981.
- [FKL88] Robert S. Fish, Robert E. Kraut, and Mary D. P. Leland. Quilt: A collaborative tool for cooperative writing. In *Conference on Office Information Systems*, pages 30–37, Palo Alto, California, March 1988.
- [Flo89] Linda Flower. Cognition, context, and theory building. *College Composition and Communication*, 40(3):282–311, October 1989.
- [Fur89] Richard Furuta. Concepts and models for structured documents. In J. Andre, R. Furuta, and V. Quint, editors, *Structured Documents*. Cambridge University Press, 1989.
- [GBL91] John D. Gould, Stephen J. Boies, and Clayton Lewis. Making usable, useful, productivity-enhancing computer applications. *Communications of the ACM*, 34(1):74–85, January 1991.
- [Gha89] M. Ghanbari. Two-layer coding of video signals for VBR networks. *IEEE Journal on Selected Areas in Communications*, 7(5):771–781, June 1989.
- [GK90] Jolene Galegher and Robert E. Kraut. Computer-mediated communication for intellectual teamwork: A field experiment in group writing. In *Proceedings of the Third Conference on Computer-Supported Cooperative Work*, pages 65–78, Los Angeles, California, 1990.

- [GNS88] David K. Gifford, Roger M. Needham, and Michael D. Schroeder. The Cedar file system. *Communications of the ACM*, 31(3):288–298, March 1988.
- [Gor80] M. D. Gordon. A critical reassessment of inferred relations between multiple authorship, scientific collaboration, the production of papers and their acceptance for publication. *Scientometrics*, 2(3):193–201, 1980.
- [Gre90] Saul Greenberg. Feasibility study of a national high speed communications network for research and development: Future applications. Technical report, Alberta Research Council, January 1990.
- [GS87] Irene Greif and Sunil Sarin. Data sharing in group work. *ACM Transactions on Office Information Systems*, 5(2):187–211, April 1987.
- [Gut85] J. Gutknecht. Concepts of the text editor Lara. *Communications of the ACM*, 28(9):942–960, September 1985.
- [H<sup>+</sup>81] Michael Hammer et al. The implementation of Etude, an integrated and interactive document production system. *SIGPLAN Notices*, 16(6):137–141, June 1981. Proceedings of the ACM SIGPLAN SIGOA Symposium on Text Manipulation.
- [Haa89] Christina Haas. How the writing medium shapes the writing process: Effects of word processing on planning. *Research in the Teaching of English*, 23(2):181–207, May 1989.
- [Hei87] Michael Heim. *Electric Language: A Philosophical Study of Word Processing*. Yale University Press, 1987.
- [HF80a] John R. Hayes and Linda S. Flower. Identifying the organization of writing processes. In Lee W. Gregg and Erwin R. Steinberg, editors, *Cognitive Processes in Writing*, pages 3–30. Lawrence Erlbaum Associates, Inc., 1980.
- [HF80b] John R. Hayes and Linda S. Flower. Writing as problem-solving. *Visible Language*, XIV(4):388–399, 1980.
- [HH76] M. A. K. Halliday and R. Hasan. *Cohesion in English*. Longmans, London, 1976.

- [HH86] Christina Haas and John R. Hayes. What did I just say? Reading problems in writing with the machine. *Research in the Teaching of English*, 20(1):22–35, February 1986.
- [HM76] J. W. Hunt and M. D. McIlroy. An algorithm for differential file comparison. Computer Science Technical Report 41, Bell Laboratories, Murray Hill, New Jersey, June 1976.
- [JC85] P. Jalote and R. H. Campbell. Atomic actions in concurrent systems. In *Proceedings of the 5th International Conference on Distributed Computing Systems*, pages 184–191, May 1985.
- [Jol89] Vania Joloboff. Document representation: Concepts and standards. In J. Andre, R. Furuta, and V. Quint, editors, *Structured Documents*. Cambridge University Press, 1989.
- [KE88] Robert Kraut and Carmen Egidio. Patterns of contact and communication in scientific research collaboration. In *Proceedings of the Conference on Computer-Supported Cooperative Work*, pages 1–12, Portland, Oregon, September 1988.
- [Knu84] Donald E. Knuth. *The T<sub>E</sub>Xbook*. Addison-Wesley, 1984.
- [KP90] Michael J. Knister and Atul Prakash. DistEdit: A distributed toolkit for supporting multiple group editors. In *Proceedings of the Third Conference on Computer-Supported Cooperative Work*, pages 343–355, Los Angeles, California, October 1990.
- [KT86] Byung G. Kim and Don Towsley. Dynamic flow control protocols for packet-switching multiplexers serving real-time multipacket messages. *IEEE Transactions on Communications*, COM-34(4):348–356, April 1986.
- [Lab87] Human Computer Interaction Laboratory. *Questionnaire for User Interface Satisfaction 5.0*. University of Maryland, 1987.
- [Lak90] Fred Lakin. Visual languages for cooperation: A performing medium approach to systems for cooperative work. In Jolene Galegher, Robert E. Kraut, and Carmen Egidio, editors, *Intellectual Teamwork: Social and Technological Foundations of Cooperative Work*, pages 453–488. Lawrence Erlbaum Associates, Inc., 1990.
- [Lam86] Leslie Lamport. *L<sup>A</sup>T<sub>E</sub>X: A Document Preparation System*. Addison-Wesley, 1986.

- [LE86] A. Lunsford and L. Ede. Why write ... together: A research update. *Rhetoric Review*, 5(1):71–81, 1986.
- [LFK88] Mary D. P. Leland, Robert S. Fish, and Robert E. Kraut. Collaborative document preparation using quilt. In *Proceedings of the Conference on Computer-Supported Cooperative Work*, pages 206–215, Portland, Oregon, September 1988.
- [LH88] Brian T. Lewis and Jeffrey D. Hodges. Shared Books: Collaborative publication management for an office information system. In *Conference on Office Information Systems*, pages 197–204, Palo Alto, California, March 1988.
- [LS83] Barbara Liskov and Robert Scheifler. Guardians and actions: Linguistic support for robust, distributed programs. *ACM Transactions on Programming Languages and Systems*, 5(3):381–404, 1983.
- [LU89] Joshua Lederberg and Keith Uncapher. Towards a National Collaboratory. Report of an Invitational Workshop at The Rockefeller University, Mar. 17–18 1989. Also available from the National Science Foundation.
- [Mai] Mainstay, Inc. *MarkUp*.
- [Mof90] Alistair Moffat. Implementing the PPM data compression scheme. *IEEE Transactions on Communications*, 38(11):1917–1921, November 1990.
- [Mor85] Robert A. Morris. Is what you see enough to get? A description of the Interleaf publishing system. In J. J. H. Miller, editor, *PROTEXT II: Proceedings of the Second International Conference on Text Processing Systems*, pages 56–81. Boole Press, October 1985.
- [NBW88] John R. Nicol, Gordon S. Blair, and Jonathon Walpole. A model to support consistency and availability in distributed system architectures. In *Proceedings of the IEEE Workshop on the Future Trends of Distributed Computing Systems in the 1990s*, pages 418–425, Hong Kong, September 1988.
- [Nel87] Theodor Holm Nelson. *Literary Machines*. Privately published, 87.1 edition, 1987.
- [Nie90] Jakob Nielsen. Traditional dialogue design applied to modern user interfaces. *Communications of the ACM*, 33(10):109–118, October 1990.

- [NKCM90] Christine M. Neuwirth, David S. Kaufer, Ravinder Chandhok, and James J. Morris. Issues in the design of computer support for co-authoring and commenting. In *Proceedings of the Third Conference on Computer-Supported Cooperative Work*, pages 183–195, Los Angeles, California, October 1990.
- [Nys86] Martin Nystrand. *The Structure of Written Communication: Studies in Reciprocity between Writers and Readers*. Academic Press, 1986.
- [Opp88] Susanna Oppen. A groupware toolbox. *Byte*, December 1988.
- [Ove82] Ray Over. Collaborative research and publication in psychology. *American Psychologist*, 37(9):996–1001, September 1982.
- [Pao82] Miranda Lee Pao. Collaboration in computational musicology. *Journal of the American Society for Information Science*, 33(1):38–43, January 1982.
- [Qui89] Vincent Quint. Systems for the manipulation of structured documents. In J. Andre, R. Furuta, and V. Quint, editors, *Structured Documents*. Cambridge University Press, 1989.
- [QV86] Vincent Quint and Irene Vatton. Grif: An interactive system for structured document manipulation. In J. C. van Vliet, editor, *Text Processing and Document Manipulation*, pages 200–213. Cambridge University Press, April 1986.
- [Rei80] Brian K. Reid. *Scribe: A Document Specification Language and its Compiler*. PhD thesis, Carnegie-Mellon University, Pittsburgh, PA, October 1980. Also issued as Technical Report CMU-CS-81-100.
- [Roc75] Marc J. Rochkind. The Source Code Control System. *IEEE Transactions on Software Engineering*, SE-1(4):364–370, December 1975.
- [Sch89] Simon Schama. *Citizens: A Chronicle of the French Revolution*. Alfred A. Knopf, New York, 1989.
- [Sel85] Robert Seliger. Design and implementation of a distributed program for collaborative editing. Master's thesis, Massachusetts Institute of Technology, September 1985.
- [Som80] Nancy Sommers. Revision strategies of student writers and experienced adult writers. *College Composition and Communication*, XXXI(4):378–388, December 1980.



- [SP88] Mike Sharples and Lyn Pemberton. Representing writing: An account of the writing process with regard to the writer's external representations. Cognitive Science Research Paper CSRP 119, School of Cognitive Sciences, The University of Sussex, 1988.
- [SP90] Mike Sharples and Lyn Pemberton. Starting from the writer guidelines for the design of user-centred document processors. Cognitive Science Research Paper CSRP 154, School of Cognitive and Computing Sciences, The University of Sussex, 1990.
- [Sti90] Gary Stix. Gigabit connection. *Scientific American*, 263(4):118–120, October 1990.
- [Sto87] Harold S. Stone. *High-Performance Computer Architecture*. Addison-Wesley, 1987.
- [Str89] Charles R. Stratton. Collaborative writing in the workplace. *IEEE Transactions on Professional Communication*, 32(3):178–182, September 1989.
- [Sub83a] K. Subramanyam. Bibliometric studies of research collaboration: A review. *Journal of Information Science*, 6(1):33–38, March 1983.
- [Sub83b] K. Subramanyam. Collaborative publication and research in computer science. *IEEE Transactions on Engineering Management*, EM-30(4):228–230, November 1983.
- [SW79] William Strunk Jr. and E. B. White. *The Elements of Style*. Macmillan Publishing Co., Inc., 1979.
- [SW90] Theodor D. Sterling and James J. Weinkam. Sharing scientific data. *Communications of the ACM*, 33(8):112–119, August 1990.
- [Tic82] Walter F. Tichy. Design, implementation and evaluation of a revision control system. In *Proceedings of the 6th International Conference on Software Engineering*, pages 58–67, September 1982.
- [VG90] Martin Vetterli and Mark W. Garrett. Joint source channel coding for real time packet services. In *Australian Video Communications Workshop*, pages 136–145, Melbourne, July 1990.
- [vRTW89] Robbert van Renesse, Andrew S. Tanenbaum, and Annita Wilschut. The design of a high-performance file server. In *Proceedings of the 9th*

*International Conference on Distributed Computing Systems*, pages 22–27, Newport Beach, California, June 1989.

- [WBMN88] J. Walpole, G. S. Blair, J. Malik, and J. R. Nicol. Maintaining consistency in distributed software engineering environments. In *Proceedings of the 8th International Conference on Distributed Computing Systems*, pages 418–425, San Jose, CA, June 1988.
- [WCL<sup>+</sup>90] Andria Wong, Cheng-Tie Chen, Didier J. Le Gall, Fure-Ching Jeng, and Kamil M. Uz. MCPIC: A video coding algorithm for transmission and storage applications. *IEEE Communications Magazine*, 28(11):24–32, November 1990.
- [Wol89] Wolfram Research, Inc. *Mathematica: For the Macintosh*, 1989.
- [Wol91] Michael F. Wolff. Computer tools: Handle with care. *IEEE Spectrum*, page 88, January 1991.

# Appendix A

## User Survey Questionnaire

### Evaluation

This section contains some questions regarding Group-Writer. If you require more space than is provided to answer a question, please continue your answer on the back of the page.

#### Part I: Personal Information

Name: \_\_\_\_\_

Age:

- \_\_\_\_\_ under 25
- \_\_\_\_\_ 25 to 35
- ☒ 35 to 45
- \_\_\_\_\_ over 45

Occupation: Admin. Assistant

Sex: \_\_\_\_\_ male ☒ female

#### Part II: Past Experience

1. Length of time you have used a Macintosh computer.
  - \_\_\_\_\_ less than 1 week
  - \_\_\_\_\_ between 1 week and 1 month
  - \_\_\_\_\_ between 1 month and 6 months
  - \_\_\_\_\_ between 6 months and 1 year
  - \_\_\_\_\_ between 1 year and 2 years
  - \_\_\_\_\_ between 2 years and 3 years
  - ☒ more than 3 years
2. How many Macintosh applications do you use regularly (i.e. that you use at least moderately frequently and with which you feel proficient)?

_____ none	<input checked="" type="checkbox"/> 4
_____ 1	_____ 5
_____ 2	_____ 6
_____ 3	_____ more than 6
3. Which Macintosh applications are you proficient in?

Microsoft Word  
Excel  
Bedford Simple Accounting

**Part III: Factors related to collaborative writing**

For this first group of questions, you are asked to rate the importance of various factors on a scale of 1 (not important) to 10 (very important).

1. *Retrieval of earlier versions of a document.* How important is it to be able to go back to an earlier version of a document that you, or your group, has created?

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Comments:

2. *Maintaining parallel versions of a document.* How important to you is it to be able to have several different versions of a document, for example, one for a conference, another for a journal?

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Comments:

3. *Comparison of document versions.* If there are several versions of a document, how important to you is it to be able to easily compare different versions?

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Comments:

4. *Annotation of a document version.* How important do you rate the ability to record comments or suggested revisions to portions of text?

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Comments:

5. *Ability to exchange files with other editors.* How important to you is compatibility with other editors, to be able to transfer Group-Writer versions to or from word processors such as Microsoft Word?

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Comments:

6. *Simple and natural to use in comparison with existing word processors.* How important is it that a group-writing system should be of similar complexity to existing word processors?

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Comments: Once you know a complex package you expect the same features and more from others!

7. *Support of electronic mail.* Should support for electronic mail be provided by a group-writing system? Should annotation be used instead of electronic mail?

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Comments:

8. *Protection of privacy, prevention of access by unauthorized people.* How important is it to you that a document you are working on be protected from unauthorized access, for example, through the use of usernames and passwords?

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Comments:

## Part IV: General Questions

For the second group of questions, please give a brief answer.

1. Would you use this kind of system in your work? Why?

Yes. Because I can review only the changed portions instead of reading all of the document to find the changes.

2. How many coauthors do you typically have when writing a paper?

It varies. So far I have had up to 15.

3. What features in a word processor which supports collaborative writing would be important to you?

The merge feature  
Annotation  
Comparison

4. What problems do you see with Group-Writer?

Although a case of too many cooks spoiling the broth. I figure this could be worked out

5. How do you think that Group-Writer could be improved? If these improvements were made, and Group-Writer was fully supported, would you be likely to use it in your work?

One thing I would like is - in the merge feature I would like the option to place the text to be added where I want in the paragraph

6. If you use Group-Writer, which features would you use regularly?

☒ Finding alternatives of paragraphs.  
☒ Merging alternative paragraphs.  
☒ Annotation of paragraphs.  
☐ The graphical Browser.  
☐ RTF Import/Export

7. Comments.

I like what I have seen so far.