JOURNAL OF ORGANIZATIONAL COMPUTING, 3(2), 195–213 (1993)

Supporting Collaboration in Digital Journal Production

Brian R. Gaines
Nicholas Malcolm
University of Calgary

As digital journals come into use there arise new possibilities for the computer support of the group processes that are involved in developing, editing, reviewing, revising, annotating, and generally using a publication. There are now a number of products and research tools designed to support group-writing teams that can be extended to support a wider range of interacting roles and activities. Most, however, require use of nonmainstream word-processing systems, and usually assume that full information is continuously available through a network to mediate and avoid conflicts. In the context of digital journals, it is more realistic to suppose that they will be distributed through both on-line and off-line media, and that a requirement for continuous network access would severely limit their use. This article reports research on group-writing tools that deviate as little as possible from conventional word processors and assume only intermittent network connection for document exchange and conflict resolution. The system developed can be used by some people as a conventional word processor, by others as a versioning and text and sound annotation system, and by others as a full hypertext system, all while working with the same corpus of documents. It offers full typographic and page-layout facilities and imports typographic text from, and exports to, the mainstream commercial word processors so that users are not locked into a nonstandard technology. It is presented here as an example of the increased functionality that may be made available through a digital journal, supporting many of the current roles and activities involved in journal creation and use while deviating minimally from current journal and word-processing practice.

Financial assistance for this work has been made available by the Natural Sciences and Engineering Research Council of Canada. The requirements specification, original version of GroupWrite, and initial user trials were developed as part of the Master's thesis research of the second author. We are grateful to Saul Greenberg and Mildred Shaw for critical comments that have improved the GroupWrite interface, and to students and colleagues who have participated in user trials for GroupWrite. The research reported here would not have been possible without access to the word-processing and page-makeup software developed by Gary Crandall, and much of the significant functionality is a direct consequence of his work.

For the sake of consistency with the first article and because the author submitted the references in alphabetical order they have been kept in that form instead of being renumbered in their consecutive order of appearance in the test as required by the IEEE style.

Correspondence and requests for reprints should be sent to Brian R. Gaines, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada T2N 1N4.

Group writing, collaboration, groupware, digital publication, electronic journals, document technology, publishing.

1. INTRODUCTION

Scholarship is a collaborative process, with researchers working together to build upon results obtained by their forerunners. In particular, modern scholarship has witnessed an increasing number of research articles being jointly authored. For example, the mean number of authors for articles 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 [22]. The percentage of single-author articles declined in the same period. However, although the joint production of documents may be seen as a significant aspect of scholarship, the availability and acceptance of systems to support it lags far behind the current capabilities of network and workstation technology. Group-writing systems to support collaborative work on document production are targeted to be part of the NSF's proposed National Collaboratory [15], a high-speed network with facilities designed to support collaboration among scholars. However, such systems are currently being explored largely in research studies and are not yet in routine use.

Whether or not collaboration occurs in the production of an article, "collaboration" occurs in the refereeing and editing activities when an article is reviewed for publication in a journal. Techniques that can support joint authors in annotating a manuscript can also be used by referees and editors to provide a critical commentary on a manuscript, and to monitor changes made by the authors in light of the comments. This type of application emphasizes the need to support collaborative editing at widely separated sites because although it is possible that collaborating authors are in the same location, it is highly likely that referees and editors will be in different locations, probably not even in the same country. A group-writing technology that supported such dispersed users would also offer the possibility of extensions to current journal capabilities by supporting digital journals that could be annotated on a continuing basis, that is, a capability for extended peer commentary.

This article reports on the development of a group-writing tool, GroupWrite, that was designed to be effective when collaborators are widely separated and have only intermittent, or no, access to communication networks. It is based on the exchange of what appear, to users, to be standard word processor files that contain, in addition to the document itself, annotations and a history of changes to the document. When several versions of a document are brought together by any means, this additional information enables them to be treated as a single corpus and the relations between them to be explored simply and naturally. We hypothesized that most potential users of GroupWrite would be current users of standard commercial word processors. They would not be ready

to accept any degradation in facilities in using an experimental system and neither would they be willing to make major changes in their work practice: certainly not in the short term. Thus, one of our major objectives was the minimalist extension of existing word-processing practice to incorporate effective support for group writing.

This article shows that such a minimalist approach is feasible and results in group-writing technology, which offers both the full functionality expected in a word processor and that necessary for version and annotation control in collaborative work, but deviating very little from current word-processing practice.

2. GROUP-WRITING SYSTEMS

Group-writing systems that have been implemented include Contexts [6], Shared Books [17], CES [24], and PREP [20]. An important precursor to all these systems is the Revision Control System (RCS) [27], which manages multiple revisions of text files, although primarily in a programming context. Through the use of an archive file, a complete record of all changes made to the document is maintained, in addition to allowing multiple parallel versions of a document to coexist. If two or more users attempt to modify simultaneously the same version, RCS gives a warning message and prevents one modification from corrupting the other. Different versions of a document can be merged, and the user is alerted if alternative versions of a portion of text exist. Differences among versions can also be easily isolated.

Several hypertext systems providing support for collaborative writing have also been implemented. ForComment [5] allows authors to attach annotations to any of the lines of text in a document. InterNote [1] extends the Intermedia environment to provide better support for annotation. It tracks the author of each annotation, and allows the contents of an annotation to be easily incorporated into the text of the main document. However, neither ForComment nor InterNote support document versioning, in which multiple versions of a document may coexist. GroupWrite, the group-writing word processor described here, is designed to manage versions at both the document and paragraph levels.

Contexts, Quilt, and PREP are hypertext-based systems that support both annotation and document versioning. Contexts [6], is an extension of the Neptune hypertext environment, allowing users their own private views or contexts of the hypertext database. Quilt [8,16] attempts to manage the complexities of collaborative writing by providing structured annotation facilities and social roles for authors. A history version of a document, complete with annotations, can be saved at any time. Using automatic paragraph comparison, readers can use history versions to see side-by-side comparisons of alternative versions of a paragraph. PREP [20] allows free-floating hypertext nodes during the planning phase of a writing project. The nodes can then be arranged in a grid, one column for the document plan, another for the document text, and the remainder for annotations. Contexts and Quilt, however, are like traditional hypertext systems

in which hypertext links disrupt the flow of the document text. PREP avoids this problem by laying out the document and the annotations in a grid. However, PREP has limited formatting capabilities, and hence writers cannot see their document in final form. GroupWrite uses a high-quality, commercially available word-processing engine to provide typographic quality formatting and output, and supplements this with hypertext annotation in the margins in such a way as not to disturb the integrity of the text.

RCS runs on Unix networks and Quilt under X-windows, and both use the editors available in the Unix environment. GroupWrite is designed to run on personal computers intermittently connected to a network and to be interface-compatible with Microsoft Word, and data-compatible with word processors having Microsoft's *Rich Text Format* (RTF) import/export.

3. DESIGN CONSIDERATIONS

In designing GroupWrite, particular attention was paid both to cognitive aspects of writing and to the way in which collaborative writing groups are structured. During the past decade, cognitive models of writing have generated much interest [21,25]. Most cognitive models view writing as an underconstrained goal-oriented activity and highlight the central role of writer intention, purpose, plans, and so on, in the composing process. In addition, efforts have been made to understand the impact of computers on the writing process and to modify cognitive models accordingly [25].

For example, Haas [11] 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. Daiute [4] found that, during a study of high school students, when revising text, use of a word processor tends to result in more word- and sentence-level revisions than when revising with pen and paper, but fewer revisions overall. Heim [13] argued that because the physical act of writing is much easier with a word processor, the resultant text will be less directed, less reflective, and less carefully formulated. When a word processor includes hypertext capabilities, with all their attendant possibilities of user disorientation and cognitive overload [2,19], the task of producing good text could become even more difficult. We have not attempted to improve the basic word processor but have made it a major design objective not to increase the cognitive load and not to disrupt the typographic structure of the text.

Stratton [26] proposed three models of the way collaborative writing groups are structured. In the first, each group member is assigned a portion of the document and is responsible for all the necessary researching, planning, drafting, and editing. In the second, one group member produces a draft of the document, which is then successively edited and revised by the other group members. In the third, each group member is assigned a nonoverlapping area of responsibility, such as research, writing, editing, or document layout. We have attempted to support all three modes of operation and allow for mixed modes

and easy interchange between them. We have assumed that for groups containing more than one writer, or containing geographically dispersed members, allowing parallel versions of a document is important. This is also helpful if one version of a document is intended for a conference, another for a journal.

Maintaining the different versions of a document as it evolves over time is also useful, both to record the flow of ideas and to allow reversion to a previous version if later changes are deemed unsuitable. The latter point is especially important for professional writers, who currently often keep paper copies of 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 examine changes made by others more easily. In addition, provision should be made so that several group members may simultaneously work on a document. To support communication among authors, annotation facilities should be provided.

Given the complexity of the writing process, it is important that a computer-based collaborative writing system be simple and natural to use, so that writers can concentrate on writing and not on the medium being used. To accommodate group members with little computing experience, different views into the document should be provided, so that the versioning mechanism can be transparent to some users.

4. GROUP-WRITING REQUIREMENTS SPECIFICATION

From the previous systems and the design considerations just discussed, it is possible to determine the significant requirements to be satisfied by a group-writing system that go beyond those of a single-user editing system [18].

1. *Reproduction of older versions.* It must be possible to trace the evolution of a document, and to revert to a previous version if subsequent changes are deemed unacceptable.
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 versions of a scientific document, 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 supplied, highlighting differences among the texts of different versions [5].
4. *Record comments and suggested revisions.* Comments and suggested revisions of portions of text made by members of a group-writing team should be recorded and easily accessible [5]. Additionally, these annotations should not disrupt the text of the original document.
5. *Compatibility with other editors.* In order for a group-writing system to be widely used, it should be compatible with popular existing editors. In a survey of 50 pairs of collaborative researchers, Kraut, Galegher, and

Egido [14] found that most had problems with incompatibilities among programs and computing environments. This often led to one partner controlling the document and incorporating the other's handwritten changes into the draft.

6. *Simple and natural to use.* The majority of potential users of a group-writing system have nontechnical backgrounds and are already familiar with commercial word processors. An acceptable group-writing system has to be very simple and natural to use, and expectations created by single-user systems such as Microsoft Word need to be supported.
7. *No requirement to manage complexity.* Users should not be required to understand and manage the versioning system. Some users may wish only to edit documents creating revisions while others in the group manage the alternative versions. For the former users the system should appear to be a conventional word processor.
8. *Support of electronic mail.* Electronic mail facilities should be integrated with a group-writing system to facilitate communication among writers.
9. *Reliably store information.* Changes to a document must be reliably stored despite computer or network failure [17].
10. *Protection of privacy.* Access by unauthorized people must be preventable, because loss or corruption of material may have financial or legal consequences [17].

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 group-writing systems. Posner and Baecker [23] recently published a study of user practices and requirements that postdates the development of GroupWrite. This study is currently the most empirically well-founded analysis of the relevant design issues, and it again emphasizes the requirements noted previously.

5. GROUPWRITE ARCHITECTURE

GroupWrite is a Macintosh program written in C+ + and its overall architecture is shown in Figure 1. The top-level functionality may be grouped into four major components:

- Document management supporting the users' concept of a single document file as the basis unit that is edited and communicated;
- Editor support through a simple and natural user interface that, as well as maintaining the content of the document and its annotation, also keeps track of what paragraphs have been changed for versioning purposes;
- Filing facilities that, as well as storing the document and annotation, also store tables of version information as separate resources in the same file as the document; and
- Interchange facilities that allow the import and export of documents in

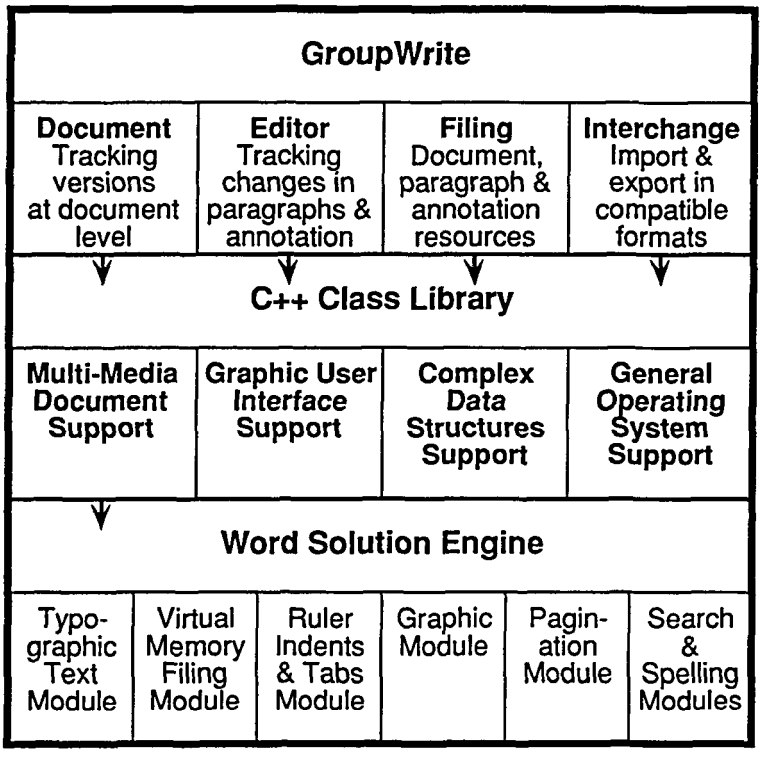

Figure 1. Overall implementation architecture

RTF format allowing users to move files between GroupWrite and commercial word processors.

The middle-level functionality is provided by an object-oriented class library written in C++, which is common to a number of developments of knowledge support systems [9] including expert systems, knowledge acquisition and hypermedia, and is not specific to GroupWrite. It may be seen as providing four classes of functionality significant to the GroupWrite implementation:

- Multimedia document support providing an open architecture word processor as a text-editing module that supports full typography, picture placement, pagination, and paragraph version tracking;
- Graphic user interface support for drawing and keyboard and mouse interaction with text-editing windows, dialogs, and interactive graphics;
- Complex data-structures support for a wide range of variable-size abstract data types including arrays of variable-size structures; and
- General operating-system support making the range of managers available in the Macintosh Toolbox available through well-structured functional classes, including filing and interapplication communication.

The multimedia document support is a very important module for GroupWrite, and the class library component is implemented at the lowest level in terms of calls to a commercial software product, Datapak's World Solution Engine [3]. This is written in M68000 assembly language for speed and provides a very comprehensive set of word-processing functions through a basic module and a series of add-on modules providing additional functionality. The main modules are:

- Typographic text module supporting display and editing of text with varying fonts and sizes;
- Virtual memory-filing module supporting the buffering of large documents to be editable within a specified memory allocation;
- Ruler indents and tabs module supporting variable paragraph width, line spacing, tab placement, and boxing;
- Graphic module supporting the placement and sizing of graphics placed in the text;
- Pagination module supporting the display and output of text pages; and
- Search and spelling modules supporting the normal word processor capabilities of searching for, and replacing, phrases, and checking the spelling and grammatical usage of words and phrases.

6. DOCUMENT AND PARAGRAPH VERSIONING

GroupWrite supports the tracking of two distinct levels of versioning: document versioning and paragraph versioning. Document versions are based on document files and a new version is created when a user opens an old version, edits it, and recommits it to the filing system as a new version; paragraph changes are automatically tracked during editing, and the new version status of a changed paragraph is recorded when the document is recommitted. Note the distinction between an edited document being *saved* for later editing, and being *recommitted* as a new version. A committed version is treated as immutable, a read-only version, such that a copy must be created for editing purposes.

The paragraph was chosen as the unit of granularity because it is both a natural conceptual unit for document producers [12], and a basic data structure in word processors. Annotation are stored in the same way as paragraphs. Each paragraph in the archive file has a unique identification (ID) number, and a link to the paragraph from which it derived. This results in a paragraph derivation tree as shown in Figure 2.

A document version is identified by a user given name, and its structure is stored as a linked list of paragraphs together with a link to the document version from which it derived as shown in Figure 3. When edited versions are recommitted, the document and the paragraph derivation orders are in correspondence, but because we also allow users to create new document versions by arbitrary reuse of existing paragraphs, no assumption is made about this relationship.

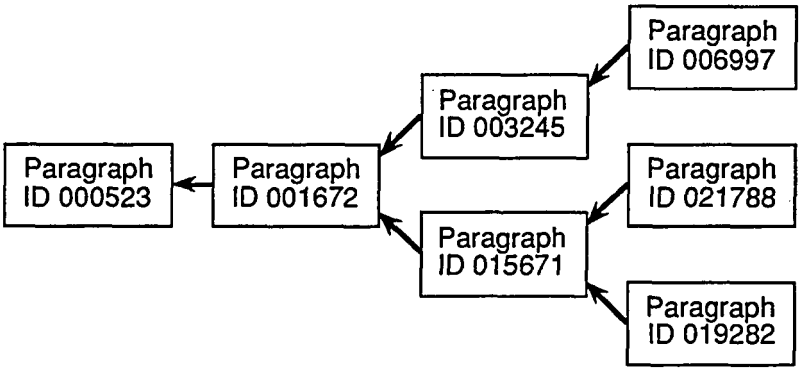

Figure 2. Paragraph derivation tree

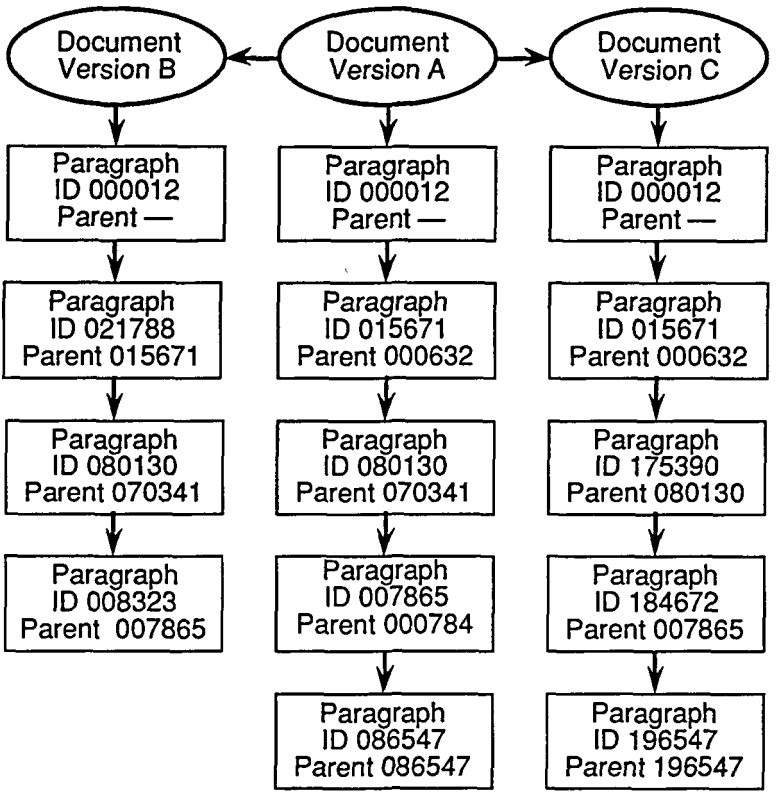

Figure 3. Document structures and derivation tree

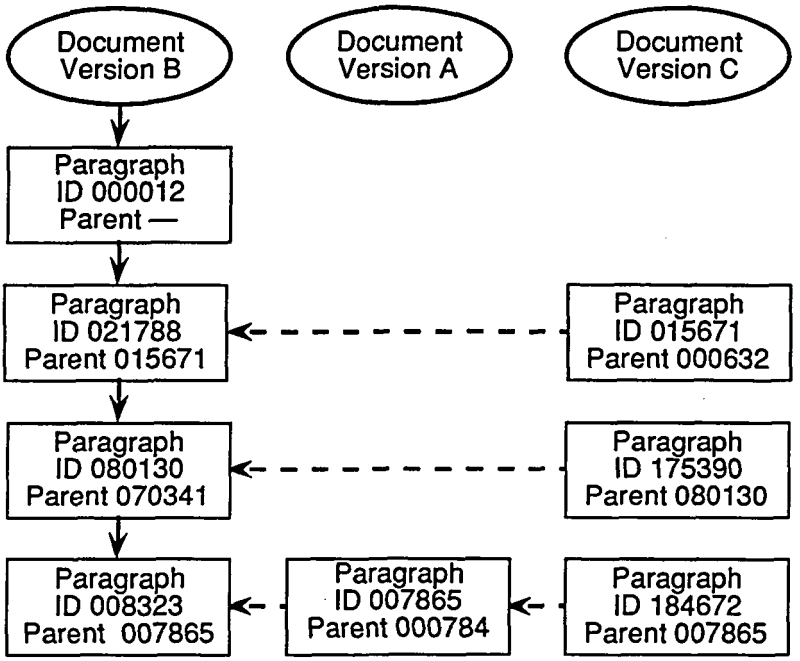

Figure 4. Alternative paragraphs (document Version B is the primary version, and version C is the secondary version, paragraphs in C with common derivations to those in B are available as alternatives; note that the paragraph in the center column is shown to illustrate the linkage and is not available during editing).

At the beginning of an editing or merging session a user selects a document version, designated the primary version. Additionally, other versions of a document may be designated secondary versions. Secondary version paragraphs that are alternatives of paragraphs in the primary version are available during editing as shown in Figure 4. They may be modified and incorporated into the primary document version. During editing, each paragraph has an associated flag to indicate whether or not it has been modified.

Figure 5 shows the internal data structures used by GroupWrite to support document and paragraph versioning and annotation.

7. USER INTERFACE DESIGN

Given the inherent complexity of hypertext, and of collaborative writing itself, the user interface to a hypertext collaborative writing system is extremely important. The critical features that must be clearly presented to the user are the existence of annotations and the relations among different versions of the document. In addition, GroupWrite is designed in such a way that the same document may be viewed in different ways depending on the expertise and the

```
Each document version is stored in a separate file and has a unique id (a string):
     Macintosh name + date/time stamp + random 32 bits
The name of each document version is the name of the corresponding file.

The data fork of the document version file contains the Datapak WSE data structures.

The resource fork of a document version file contains the following (resource types
given in parentheses):
     version of GroupWriter used (vers)
     unique ids of previous document versions (STR#).
     names of previous document versions (STR#)
     text annotations (STR#)
     sound annotations (SND)
     name of creator of the document version (STR)
     paragraph records
     annotation records

(Note that a STR# resource consists of a list of strings.)

Each paragraph has a unique id that is 32 bits long.  The high order 16 bits identifies
the STR# string containing the uid of the document version for this paragraph.  The low
order 16 bits is the index of the paragraph in the original document version in which this
version of the paragraph first occurred.

The following structure is used for paragraphs ids:
     typedef struct {
           int    uidDoc;
           int    index;
     } idPara;

Paragraph records are stored as an array, with one record for each paragraph in the
document:
     typedef struct {
           idPara id;
           int    annIndex;     /* index into annotation array */
           long   beginPtr;     /* pointer to beginning of para in document text */
           short  nPrev;        /* number of previous versions of this para */
           idPara prev[];       /* uid's of paragraph's ancestors */
     } paraInfoR;

Annotations are stored in a single array.  The next field gives the array index of the
next annotation for the current paragraph, or is -1 if the current paragraph has no more
annotations:
     typedef struct {
           Str31  ptitle;
           Str31  pcreator;
           long   creationDate;
           Byte   kind;         /* text or sound */
           int    id;           /* identifies string in STR# or a SND */
           int    next;
     } annRec;
```

Figure 5. Versioning data structures

inclination of the user. It may be used naively as a conventional word processor, as a word processor with versioning and annotation facilities, or as a structured hypertext system. This allows a group to arrange itself such that some writers can create and edit document versions conventionally while others take responsibility for reconciling versions.

Figure 6 illustrates the user interface to the multimedia document editing class and some of the underlying capabilities of Word Solution Engine. The interface is designed to be very similar to those of commercial word processors with which users are familiar. Support for managing document versions is also structured such that users can use facilities similar to those in conventional word

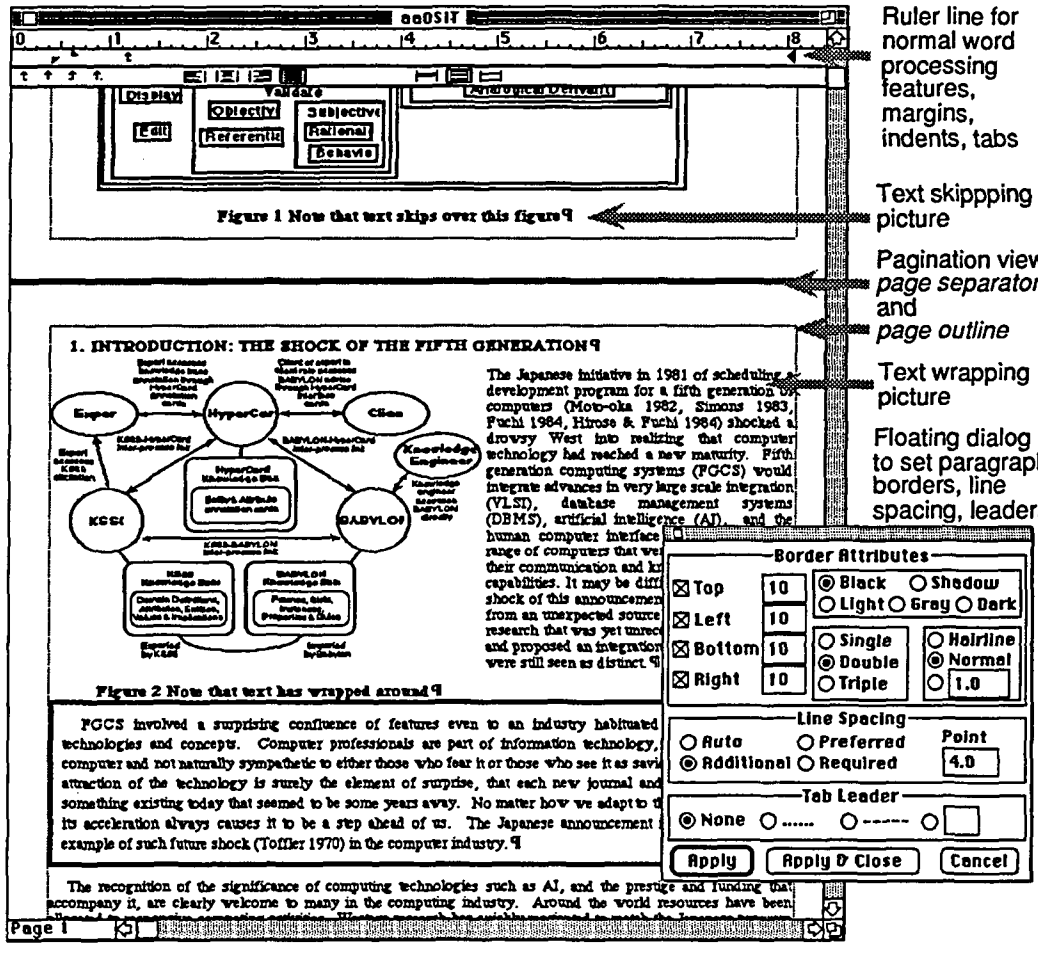

Figure 6. Word Solution Engine features

processors. They can track versions through naming conventions if they wish and open both primary and secondary versions through a standard file-opening dialog. However, a graphic browser showing relationships among document versions is also provided as shown in Figure 7. This display is created dynamically by examination of the document version information in the files within the working folder on the Macintosh when the browser is selected. Thus, it represents whatever files the user has placed in the folder and the version relations, if any, among them. Double clicking on a document version opens it for editing as the primary document version, with any other selected versions being opened as secondary versions.

The primary document version appears in a normal edit window as shown in Figure 8 with lines in the left margin indicating the existence of alternative paragraphs or annotations. The appearance of the line varies depending on what types of annotation are present, and whether alternative paragraphs are

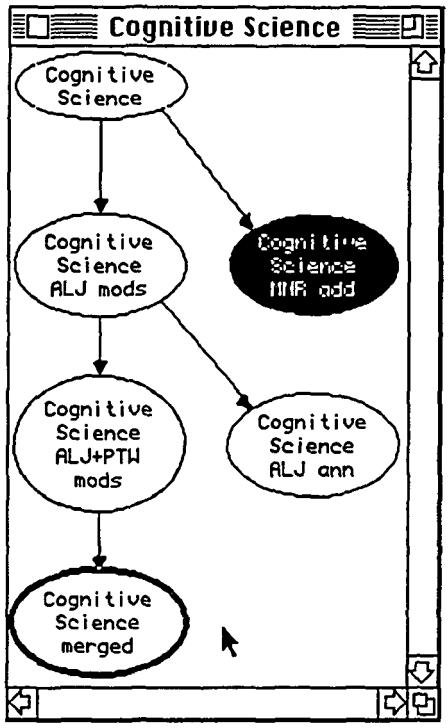

Figure 7. Document version browser

present. The mouse pointer changes shape when moved over the line to indicate that a popup menu is available. This menu may also be accessed by holding down the option key while clicking the mouse in the marked paragraph. The popup menu provides a number of operations that may be preformed on annotations and alternative paragraphs. They may be displayed in a separate window by the *Show* operation; inserted into the document text by the *Place Before, Place After,* or the *Replace* operations; removed by the *Remove* operation. Sound annotations may be heard by selecting the *Play* operation. Annotations and alternative paragraphs are selected from the submenu, with annotations appearing below the dotted line.

8. MANAGING COLLABORATION

One of the unusual features of GroupWrite compared with other collaborative writing systems is that those collaborating are not assumed to be coordinated through a network as they work on a document. This was a deliberate design decision to reflect the reality of the way in which most joint articles are produced, that is, by coauthors at different sites, working at different times, and often working on the same version of the document at different times. The automatic version tracking in GroupWrite allows the resultant documents' rela-

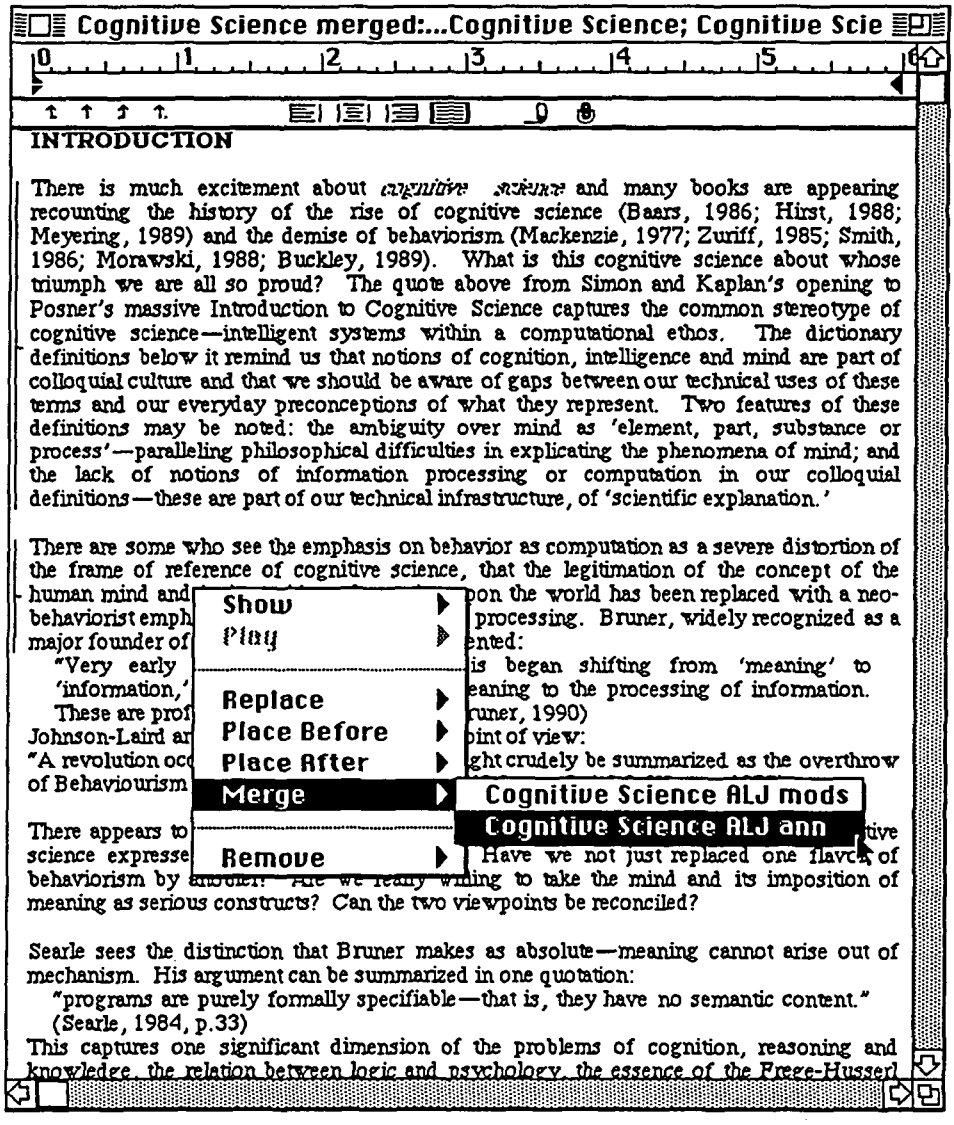

Figure 8. Alternative paragraph markers and the popup menu

tionships to be made clear to the coauthors. The relationship among documents is shown through the graphic version browser as shown in Figure 7, and potentially conflicting versions of a paragraph are shown and made accessible through the side markers and popup menus as shown in Figure 8. However, one needs to go beyond this to support conflict resolution among documents and coordination among coauthors. GroupWrite has two features designed to support these processes, a paragraph comparison and merge capability, and a text and voice annotation capability.

In Figure 8, when *Merge* is selected in the popup menu, alternative paragraphs and textual annotations are compared on a sentence-by-sentence basis with the corresponding paragraph in the document version being edited, and the differences displayed in a separate window as shown in Figure 9. Differing sentences are marked by brackets in the left margin, the thick bracket indicating that the sentence originated in the document version being edited. Sentences to be included in the merged paragraph are selected by clicking in the corresponding bracket. The merge is completed by double clicking in one of the brackets, at which time the merged paragraph is inserted into the document version being edited. This is a fairly primitive facility, and one can envision systems with more intelligent support of document version merging being developed. However, it is not apparent yet what will be useful in practice, neither too simplistic to be useful nor too complex to be usable, and this should become a major area of research in group-writing system design.

We have assumed that much coordination among coauthors will take place through normal mechanisms such as meetings, telephone, and email that are outside the technical scope of GroupWrite itself. However, it also important that commentaries and annotations may be attached to specific items within the text and this has been an intrinsic part of the design. Paragraphs may be given either text or sound annotations through dialogs such as that shown in Figure 10. Annotations are created by placing the cursor within the paragraph to be annotated and either selecting the appropriate menu item or clicking on the appropriate icon in the ruler of Figure 8. Thus, coauthors do not have to create alternative versions of paragraphs for later merging. They can instead, or as well, make comments to those primarily responsible for material that should be changed in some way, and leave it to the originator to decide how to make those changes.

The system has also been designed to integrate well with electronic mail. The files created by GroupWrite appear as normal Macintosh files and a single document file contains all the information pertaining to it, including content,

There are some who see the emphasis on behavior as computation as a severe distortion of the frame of reference of cognitive science, that the legitimation of the concept of the human mind and its imposition of meaning has been replaced with a neo-behaviorist emphasis on mindless information processing. Bruner has commented:

There are some who see the emphasis on behavior as computation as a severe distortion of the frame of reference of cognitive science, that the legitimation of the concept of the human mind and its imposition of meaning upon the world has been replaced with a neo-behaviorist emphasis on mindless information processing. Bruner, widely recognized as a major founder of cognitive science, has commented:

Figure 9. Merging two paragraphs

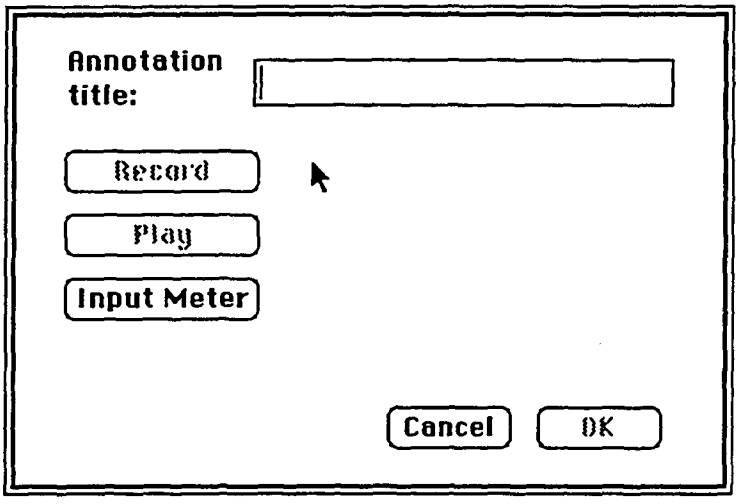

Figure 10. Sound annotation

annotation, and versioning. GroupWrite files may be distributed on disks or sent through local area networks or through Internet using existing protocols. In particular, GroupWrite documents may be sent through electronic mail as Eudora [7] *attachments*, and hence both local and remote authors may communicate messages and document versions through a single uniform system. GroupWrite documents may also be imported and exported in RTF format, which encodes full typography and layout, including embedded pictures, and is an interchange format for the main commercial word processors. This export capability also enables GroupWrite documents to be read on platforms other than the Macintosh, although the version information will not be available.

The variety of facilities for coordination provided by GroupWrite makes it possible for coauthors to operate in many different modes. Some may act to coordinate versions whereas others work only on one version. Some may act as editors whereas others act largely as commentators. Some may exchange material on disk through the mail, and others may coordinate through email and document attachment. These different roles and approaches may be mixed and may change as the joint article develops. The following is an example of one possible scenario.

1. Coauthor A outlines an article in draft form (V1) and sends a copy on disk through the mail to coauthor B.
2. Coauthor B edits and extends the article (V1.1) and sends a copy through email to coauthor C.
3. Coauthor C edits and extends the article (V1.1.1) and sends a copy back through email to coauthor B.
4. Coauthor A continues to work on the original article (V1.2) and sends a copy on disk through the mail to coauthor B.

5. Coauthor B opens C's version (V1.1.1) in the context of his original (V1.1) and A's amended version (V1.2), and creates a merged version (V1.1.1.1) which he sends to both A and C with annotation asking them to make final changes.
6. Coauthor B receives back A's final version (V1.1.1.1.1) and B's final version (V1.1.1.1.2), both annotated with requests for final checks and amendments, opens one in the context of the other, creates a merged version, checks all the annotations, opens the merged version in the context of all the previous versions and checks all variants for possible improvements, and then prints out the article to be sent to a journal.

In this scenario, coauthor B has acted as overall coordinator using the versioning and merging facilities, and both A and C have used GroupWrite as an ordinary word processor with annotation capabilities. Compared with current practice using ordinary word processors, B's task has been made very much easier because the changes made by all three coauthors in parallel are being tracked automatically and made visible through GroupWrite.

9. CONCLUSIONS

Current networking, workstation, and software engineering technologies make it feasible to develop word processors supporting collaborative writing processes and digital journals. It is feasible to offer the full functionality required with minimal extensions to current word-processing user interfaces and requiring little change in work practice. This is an effective way of supporting a versioning and annotation system for the production of documents by a group of writers. It can also support the refereeing and editing processes of a digital journal, and the extension of the journal publication process by annotation and amendment of a publication in later versions.

It is appropriate, in conclusion, to return to the list of requirements given in Section 4, and discuss to what extent GroupWrite satisfies them.

1. *Reproduction of older versions.* GroupWrite makes it possible to trace the evolution of a document, and to revert to a previous version if subsequent changes are deemed unacceptable.
2. *Maintaining multiple parallel versions of a document.* GroupWrite supports the existence of multiple parallel versions of a document.
3. *Comparison of versions.* GroupWrite supports the comparison of alternative and previous versions of a document, highlighting differences among the texts of different versions.
4. *Record comments and suggested revisions.* GroupWrite supports text and voice annotation in a way that does not disrupt the text of the original document.

5. *Compatibility with other editors.* GroupWrite is designed to be very similar in operation and functionality to commercial word processors.
6. *Simple and natural to use.* GroupWrite provides version tracking through a simple user interface that involves minimal changes to that of normal word processors.
7. *No requirement to manage complexity.* GroupWrite performs version management automatically even when being used as a conventional word processor.
8. *Support of electronic mail.* GroupWrite, in its current version, does not incorporate electronic mail, but GroupWrite documents may be sent as email attachments.
9. *Reliably store information.* GroupWrite stores documents as self-contained files and does not rely on central servers or additional data structures containing linkage information.
10. *Protection of privacy.* GroupWrite, in its current version, makes no special provision for document security, but lack of reliance on a central server aids security, and other programs may be used to encrypt and authenticate GroupWrite files on disk and in transit if required.

The development of GroupWrite is part of a program of research on knowledge support systems [9] to support human knowledge processes at a wide range of levels from electronic communication, through document production, to hypermedia, knowledge modeling, and reasoning. In particular, the document production environment is seen as a base level into which other support systems must be integrated in order to support fully the knowledge processes of the scientific community. For example, we recently reported on a word processor similar to GroupWrite that is in itself a full expert system shell providing a visual language for formal knowledge structures through, what appears to be, embedded diagrams in a document [10].

There is now a potential in digital document production and dissemination both to extend existing document systems with enhanced functionality and to provide them with new capabilities that go beyond any existing functionality. Digital documents emulate an existing medium, but they are also transforming it into a new medium whose potential depends largely on our creative imaginations and their effective application both to support and to develop user requirements.

REFERENCES

[1] T. Catlin, P. Bush, and N. Yankelovich, "InterNote: Extending a hypermedia framework to support annotative collaboration," in *Hypertext' 89 Proc.*, 1989, pp. 365–378.

[2] J. Conklin, "Hypertext: An introduction and survey," *IEEE Computer*, vol. 20, no. 9, pp. 17–41, 1987.

[3] G. Crandall, *Word Solution Engine Programmer's Manual*, Vancouver, WA: DataPak, 1990.

[4] C. Daiute, "Physical and cognitive factors in revising: Insights from studies with computers," *Research in the Teaching of English*, vol. 20, no. 2, pp. 141–159, 1986.

[5] R. Dalton, "Group-writing tools: Four that connect," *Information Week*, pp. 62–65, March 9, 1987.
[6] N.M. Delisle and M.D. Schwartz, "Contexts—A partitioning concept for hypertext," *ACM Trans. on Office Information Systems*, vol. 5, no. 2, pp. 168–186, 1987.
[7] S. Dorner and C. Gedney, *Eudora Manual*. San Diego: Qualcomm (anon ftp ftp.qualcomm.com), 1991.
[8] R.S. Fish, R.E. Kraut, and M.D.P. Leland, "Quilt: A collaborative tool for cooperative writing," in *Proc. of Conf. on Office Information Systems*, 1988, pp. 30–37.
[9] B.R. Gaines, "Knowledge support systems," *Knowledge-Based Systems*, vol. 3, no. 3, pp. 192–203, 1990.
[10] B.R. Gaines and M.L.G. Shaw, "Documents as expert systems," in *Research and Development in Expert Systems. Vol. 9: Proceedings of British Computer Society Expert Systems Conference*, M.A. Bramer and R.W. Milne, Eds. Cambridge, UK: Cambridge University Press, 1992, pp. 331–349.
[11] C. Haas, "How the writing medium shapes the writing process: Effects of word processing on planning," *Research in the Teaching of English*, vol. 23, no. 2, pp. 181–207, 1989.
[12] M.A.K. Halliday and R. Hasan, *Cohesion in English*. London, UK: Longmans, 1976.
[13] M. Heim, *Electric Language: A Philosophical Study of Word Processing*. New Haven, CT: Yale University Press, 1987.
[14] R.E. Kraut, J. Galegher, and C. Egido, "Relationships and tasks in scientific research collaboration," *Human–Computer Interaction*, vol. 3, pp. 31–58, 1987.
[15] J. Lederberg and K. Uncapher, "Towards a National Collaboratory," in *Rep. of an Invitational Workshop at The Rockefeller University*, 1989.
[16] M.D.P. Leland, R.S. Fish, and R.E. Kraut, "Collaborative document preparation using Quilt," in *Proc. of Conf. on Computer-Supported Cooperative Work*, 1988, pp. 206–215.
[17] B.T. Lewis and J.D. Hodges, "Shared books: Collaborative publication management for an office information system," in *Proc. of Conf. on Office Information Systems*. 1988, pp. 197–204.
[18] N. Malcolm, "GroupWriter: A Word Processor for Collaborative Document Preparation," University of Calgary, Calgary, Canada, 1991.
[19] C. McKnight, A. Dillon, and J. Richardson, *Hypertext in Context*. Cambridge, UK: Cambridge University Press, 1991.
[20] C.M. Neuwirth, D.S. Kaufer, R. Chandhok, and J.J. Morris, "Issues in the design of computer support for co-authoring and commenting," in *Proc. of 3rd Conf. on Computer-Supported Cooperative Work*, 1990, pp. 183–195.
[21] M. Nystrand, *The Structure of Written Communication: Studies in Reciprocity Between Writers and Readers*. New York: Academic, 1986.
[22] R. Over, "Collaborative research and publication in psychology," *American Psychologist*, vol. 37, no. 9, pp. 996–1001, 1982.
[23] I.R. Posner and R.M. Baecker, "How people write together," in *Proc. of 23rd Annual Hawaii Int. Conf. on Systems Sciences*, 1992, pp. 127–138.
[24] R. Seliger, "Design and implementation of a distributed program for collaborative editing," Massachusetts Institute of Technology, Cambridge, MA, 1985.
[25] M. Sharples and L. Pemberton, "Representing writing: An account of the writing process with regard to the writer's external representations," School of Cognitive Sciences, The University of Sussex, 1988.
[26] C.R. Stratton, "Collaborative writing in the workplace," *IEEE Trans. on Prof. Comm.*, vol. 32, no. 3, pp. 178–182, 1989.
[27] W.F. Tichy, "Design, implementation and evaluation of a revision control system," in *Proc. of 6th Int. Conf. on Software Engineering*, 1982.