December 2002

DESIGN REQUIREMENTS FOR COLLABORATIVE WRITING TOOLS THAT SUPPORT DISTRIBUTED, INTERNET-BASED WORK

Paul Lowry
Brigham Young University, Marriott School of Management

DESIGN REQUIREMENTS FOR COLLABORATIVE WRITING TOOLS THAT SUPPORT DISTRIBUTED, INTERNET-BASED WORK

Paul Benjamin Lowry
Brigham Young University, Marriott School of Management
Paul.Lowry@BYU.Net

Abstract

This paper proposes requirements necessary for designing future collaborative writing tools to support distributed work teams on the Internet. Requirements are proposed from literature, and then compared to previous and existing CW tools such as Microsoft Word™. Because no tool adequately addresses the needs of distributed CW teams, a new CW tool called Collaboratus is proposed and reviewed.

Introduction

Collaborative writing (CW) is a critical form of professional communication that can be more effective than single-author writing (Gere & Abbott, 1985) and is ubiquitously performed in industry and academia (Couture & Rymer, 1989; Ede & Lunsford, 1990; Gordon, 1980). Furthermore, while CW is already pivotal to many group processes, it is likely to increase in importance and complexity because of the increased use of distributed work, driven by increased globalization, competition and Internet usage. While CW is widely performed and increasing in importance, current writing tools provide insufficient support. Indeed, little has changed since 1994 when Galegher and Kraut asserted: "…the conventional word processor, designed for individual use and highly evolved for layout and production of paper documents, needs to be rethought and recast as a group writing tool" (Galegher & Kraut, 1994). Anecdotal evidence that word processors are inadequate for distributed CW is derived by the proliferation of protocols designed to help groups write, but which undermine group awareness (Galegher & Kraut, 1994).

To help develop a new generation of Internet-based CW tools, this paper presents design requirements for building such tools. These requirements are compared against Microsoft Word™ and previously produced CW tools, which shows the limitations of these tools for CW. Finally, a new CW tool, Collaboratus is presented that is proposed to best adhere to these requirements.

CW Tool Requirements

This section present CW tool requirements, based on a review of literature; in terms of work modes, strategies, activities, communication, and technology.

Work modes: CWS should support all major collaborative work modes, which requires these four modes be treated differently in collaborative systems design (Schlichter, Koch, & Burger, 1997). Thus, CWS and interfaces must be designed with "asynchronous awareness" to support communication across space and time (Neuwirth, Morris, Regli, Chandhok, & Wenger, 1998).

Strategies: CW tools need to support all major writing strategies (Posner & Baecker, 1992). Parallel partitioned writing capability may be particularly important, because it allows task decomposition so that team members can work at the same time with better coordination (Moon & Sproull, 2000).

CW Activities: CWS should support all the major CW activities and should to ensure seamless transitions between activities and to allow activities to be chosen in any order (Posner & Baecker, 1992). Additionally, CWS must allow multiple users to perform various writing activities at the same time on the same tool (Baecker, Nastos, Posner, & Mawby, 1993). In addition,

CWS systems need flexibility and to not enforce high levels of structure (Erickson, 1989). CWS need to support the major CW activities while remaining intuitive, so as not to distract from the CW process (Adkins, Reinig, Kruse, & Mittleman, 1999).

Brainwriting: CWS must support communication and planning through brainstorming tools (Horton, Rogers, Austin, & McCormick, 1991). Similarly, to support planning and outlining, CWS should allow collaborators to generate idea collections to categorize and store brainstorming output (Kraut, Galegher, & Egido, 1988).

Planning: CWS need to support initial group planning, brainstorming, and document outlining (Horton et al., 1991). Additionally, because planning typically is not complete before writing commences, CWS should support communication about planning (Neuwirth, Kaufer, Chandhok, & Morris, 1990), which includes negotiation support to resolve differences of opinion on group plans (Beck, 1993) and facilitating discussions involving project scheduling, courses of action, and goals (Posner & Baecker, 1992).

Writing: Most CWS tool designers intentionally do not support advanced word processing features in their tools, with the philosophy that draft documents can be easily exported to a word processor where one individual can perform the copyediting. Thus, the most basic requirement for supporting the CW activity is to provide word processing (Baecker et al., 1993). Additionally, to allow collaborators to recover from editing mistakes, CWS should have reversible execution / undo (Ellis, Gibbs, & Rein, 1991) on multiple levels (Baecker et al., 1993). In addition, collaborators need concurrency control and conflict resolution to resolve discrepancies between participants' simultaneous operations, such as text-section locking (Ellis et al., 1991).

Reviewing: The primary requirement for supporting the reviewing activity is to provide document annotations (Neuwirth et al., 1990), which are annotative comments made by reviewers. Annotations should be improved by having annotations that point to specific pieces of text so reviewers can easily discern the passage to which a specific annotation applies (Miles, McCarthy, Dix, Harrison, & Monk, 1993).

Roles: CWS must support multiple collaboration roles (Neuwirth et al., 1990). Likewise, each participant's role must be explicitly defined at the outset of collaboration (Posner & Baecker, 1992). Moreover, CWS need to allow a flexible number of participants (Ellis et al., 1991). CWS must preserve collaborator identities (Posner & Baecker, 1992), which means every member of a CW team should know who is playing what role and to be able to see the specific content contributions a given person is making.

Because CW is dynamic and iterative, CWS must allow for shifting roles throughout the CW process (Neuwirth et al., 1990). Users should be allowed to easily create, join, or leave CW sessions at arbitrary times without significant disruption to the CWS session (Beck, 1993).

Interpersonal communication: CWS need to support social interactions (Posner & Baecker, 1992) and communication about project scheduling and plans (Neuwirth et al., 1990). Interpersonal communication is not the only form of communication that needs to be supported; in addition, writers need to coordinate their work efforts to improve their group awareness. Specifically, some type of signaling and messaging is critical for communication, or breakdowns in understanding will occur (Dubs & Hayne, 1992).

Information access: To increase information access, CWS should provide access to relevant information (Posner & Baecker, 1992). One approach to support this requirement is to allow creation of separate document segments (Posner & Baecker, 1992), a feature that allows users to access the document sections with which they are concerned. Although CWS should support different document segments, users should still have access to the entire document (Posner & Baecker, 1992). However, CWS should allow several document access methods according to a user's role and responsibilities: such as write, annotate, and read (Posner & Baecker, 1992). Finally, participants should be able to access a group document concurrently or sequentially (Ellis et al., 1991).

Information views: Another way to enhance information access is to provide different views of collaboration artifacts. Specifically, a CWS tool's interface should depict overall group activity with minimal complexity and distraction (Ellis et al., 1991). An outline view should be used to represent document sections in a hierarchical structure (Kraut et al., 1988). Outlines not only served an organizational purpose, but also help serve the requirement of authors needing to be able to break a document down into more manageable "chunks" (Kirby & Rodden, 1995).

Version control: Providing version control is another requirement to enhance information access and information views (Kraut et al., 1988). Many situations exist when users need to know the changes that have occurred to a document (Kraut et al., 1988; Miles et al., 1993). CWS must ensure that all users access the same version of the shared document (Baecker et al., 1993) and

that the accessed version is the most recent version available (Ellis et al., 1991). Additionally, CWS should allow revisions to exist as distinct drafts (Neuwirth et al., 1990). However, version control should not just include tracking different versions of text, it should also track creation author, what was changed, when changes were made (Baecker et al., 1993).

External information: CWS need to provide support for meta-textual information such as idea collections, outlines, plans, annotations, and comments (Kraut et al., 1988), all of which may be provided from external sources. Additionally, CWS should store and display external representations that are a part of normal writing process (Neuwirth et al., 1990). A critical piece to this support is the ability to input/import and output/export of data in a variety of formats or standard protocols (Neuwirth et al., 1990), including basic printing capabilities (Adkins et al., 1999).

Technology support: CWS technology must address compatibility issues with software and hardware (Neuwirth et al., 1990); in particular, it should support heterogeneous hardware, software, and networks (Galegher & Kraut, 1994). Likewise, it must and have easy installation and support (Adkins et al., 1999). CWS technology must have protocols that account for the differing requirements of various media (e.g. audio vs. video) (Ellis et al., 1991). Ultimately, a seamless integration should exist in exchanging data with other technologies (Baecker et al., 1993), such as Word™ and PowerPoint™. CWS also needs to be responsive (Ellis et al., 1991), robust, (Ellis et al., 1991), and flexible (Kraut et al., 1988). Responsiveness means that CWS needs to have short response times and short notification times (Ellis et al., 1991). CWS should allow different rights for specific participants on different document sections (Neuwirth et al., 1990).

In terms of synchronous communication, CWS needs to provide real-time interaction and notification, including support for multicasts (Ellis et al., 1991). For asynchronous communication, low-band width communication should be provided (Ellis et al., 1991). Finally, since group members should have instantaneous access to the latest document, distributed replication techniques should not be used (Ellis et al., 1991), since replication can cause serious version control and timing problems.

Other Tools

Given the proposed requirements for distributed, CW tools, this section review Microsoft Word™ and other CW tools against the requirements.

Word™: Word 2000™ has made a distinct improvement from Word™'97 in supporting CWS features; however, Word™ suffers from the overall limitation in that it was designed as a proprietary, single-user tool that had group features added to it after the initial, single-user design was created. As a result, Word™ was not properly designed for complex group awareness, coordination, and distributed CW. Furthermore, too many of its group-oriented features poorly integrate with each other. Worse, with its massive installation size and non-support of Java, Word™ cannot support large groups or provide low-bandwidth communications support required for distributed CW using Internet browsers. Table 1 compares the group features of Word™ to some of the critical requirements of CW tools.

Table 1. Word™ Group Features Compared to Requirements

Word™ "group" feature	Supports F2F?	Supports Synch. different place?	Supports Asynch. same place?	Supports Asych. different place?	Supports large groups?	Supports low bandwidth, browser-based communication?
Annotations/ comments	⊙	○	⊙	⊙	○	○
Change-tracking and merging documents	○	○	⊙	⊙	○	○
Versioning	⊙	⊙	⊙	⊙	○	○
Master documents	⊙	○	○	○	○	○
Email/routing	⊙	⊙	⊙	⊙	⊙	○
Discussions	⊙	●	●	●	⊙	○
Document subscriptions	⊙	⊙	⊙	⊙	○	○
NetMeeting integration	⊙	⊙	○	○	○	○

Key: ○ = no support ⊙ = partial support ● = full support

Given these results, some researchers have wondered whether or not future CW tool should be custom built, or if it is reasonable to extend the features of Word™ through the use of programming API's. However, given the massive number of functionality gaps in Word™ extending Word™ through API's is not a viable approach, because Word™ would have to be completely re-written.

Even if a researcher had the resources to engage in a monumental re-write of Word™, or to extend it via API's, several other limitations would greatly impede this approach: (1) Word™ does not support Java and thus cannot be run through a browser to support asynchronous teams, large teams, and low-bandwidth communications. (2) Additionally, Word™ has a massive software footprint on a client computer that requires 147 MB of available hard-disk space on the average, 36 MB of RAM, and a Pentium 75 MHz processor. These client-side requirements dramatically increase when installing the required software that provides group support (e.g. Server Extensions, NetMeeting™, Outlook™); thus, an appropriate installation requires 1 Gig of hard drive space, 128 MB of RAM, and a Pentium 400 MHz processor, or higher. Thus, these requirements undermine its ability to support Internet devices, low-end computers, and wireless devices. Administration also becomes unwieldy for any but the smallest of groups. (3) Word™ and all of its supporting technologies are proprietary to the Windows™ operating system, which precludes multi-platform support. (4) Word™ was built as a single-user tool with group-oriented extensions. Thus, it has not been optimized for scalability and distributed architectures needed to support asynchronous work, synchronous remote work (e.g. multi-casts), and large groups. (5) Word™ API's place researchers at the control of Microsoft for support, upgrades, and bug fixes. Additionally, future Microsoft upgrades will likely invalidate past approaches and introduce new bugs, limitations, challenges, and conflicting features. Table 2 summarizes the limitations of extending Word™ as a CW tool.

Table 2. The Limitations of Word™ for CW

Limitation	Severity	Ability for third party to work around limitation
Lack of Java support	Extremely high	Extremely low; can use Citrix server, but still will not support low-band width
Large footprint / resource consumption	Low for PC-based groups; moderate for notebook-computer groups; extremely high for thin and Internet clients	Cannot be reasonably addressed
Proprietary technology	High	Cannot be reasonably addressed
Not scalable or able to support distributed architectures	Low for small groups, high for large groups	Cannot be reasonably addressed
Lack of control of upgrades and software development	Extremely high	Extremely low to moderate; can attempt to influence through Microsoft support and through beta release programs

Previous CW tools: While many notable CW tools have been produced, these tools do not adequately support the requirements for distributed CW. Examples of notable past attempts to create CW tools include **GROVE** (Ellis et al., 1991), **Quilt** (Fish, Kraut, Leland, & Cohen, 1988), **SASSE** (Baecker et al., 1993), **Collaborwriter** (McAlpine & Golder, 1994), **Collaborative Editing System (CES)** (Greif, Seliger, & Weihl, 1986), **ShrEdit** (Olson, Olson, Storrosten, & Carter, 1992), **Iris** (Koch, 1995). Version control systems, protocols such as **CVS**, and document management systems also tend to fall short of full CW support, especially in terms of group awareness and simultaneous coordination.

Because Word™ and the other listed tools were deemed to not support sufficiently the requirements of CW, researchers at the Center for the Management of Information (CMI) at the University of Arizona set out to build a new CW tool, called Collaboratus, which would start to address the requirements for distributed CW on the Internet. Collaboratus was built using Nunamaker's System Development Methodology (SDM) (Nunamaker Jr., Chen, & Purdin, 1991), which is the result of years of field and laboratory research, as described in (Lowry, Albrecht, Nunamaker Jr., & Lee, 2002).

A New CW Tool: Collaboratus

Collaboratus is different from previous CW tools, because it is the first tool that has been designed from the ground up to take advantage of the Internet and Java technologies. This is an important advance, because it allows Collaboratus to run through platform-independent web-browsers on any platform that can run a Java virtual machine (VM). Users can dial-in via modem

connections, or they can be connected directly at higher data transmission rates on a local area network. Additionally, Java support has allowed a flexibility and scalability that can better support large, distributed teams.

One of the important design decisions of Collaboratus was to not support data replication, as is found in Lotus Notes™. In Notes™, users can work on a task without being connected to the network and when they connect to the network, their work is automatically "replicated" to the database and shared with other users. Replication technology is extremely complex and prone to replication conflicts; thus, it can be particularly problematic with collaborative writing, because it disallows users from knowing what contemporaneous changes are occurring with a document, unless group members continually replicate their changes to a central network (an impractical solution for remote workers on low-speed connections). Thus, such distributed replication can greatly undermine group awareness of CW processes. As such, the decision was made to only support a server-oriented architecture where users must be connected to the server in order to work. Additionally, The Collaboratus server has been optimized for low-bandwidth clients and has a small, 155K common download for full-featured applications. Support for dial-up clients has been successfully achieved because the messaging system is highly efficient and only uses small communication packets -- the server performs most of the work. The server also supports asynchronous and synchronous groups.

The underlying server technology on which Collaboratus is based is the CMI Collaborative Server Architecture, which allows rapid, object-oriented development of collaborative, highly interaction applications. As a result, Collaboratus has a powerful server that can be easily setup and administered. Because the architecture is Java-based, the client applications can easily be installed locally or run directly from web pages from any major web browser. Additionally, the Collaboratus server supports a pure TCP/IP implementation, it works within firewalls, it is highly scalable, and requires no database configuration or extensive DBMS support. The Collaboratus server takes approximately 30 seconds to install. A given Collaboratus administrator is able to quickly check the status of the server, start and stop the server, log the activity that is occurring the given session, and move the server to run on different sockets. Additionally, an administrator can automatically save the session data to an archive log, which can be backed up and restored for later use. The architecture is fully described in (Albrecht, 2000).

The integrated applications that compose the Collaboratus tool include: GroupAdministrator, GroupAgenda (Figure 1), GroupOutliner, GroupVoter (Figure 2), GroupBrainstormer, GroupCommenter, GroupCategorizer, GroupAnalyzer, GroupSketcher, and GroupWriter, as further explained in (Lowry, 2002).

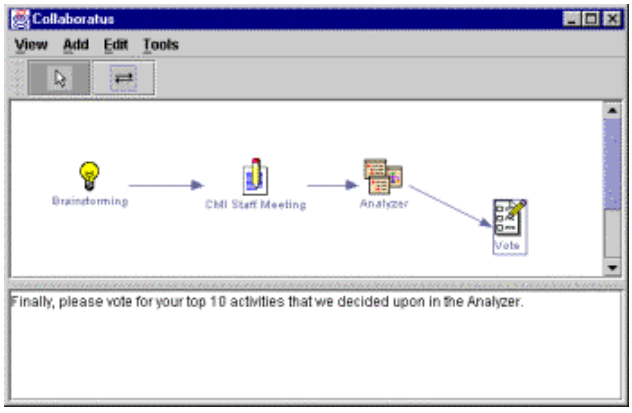

Figure 1. GroupAgenda

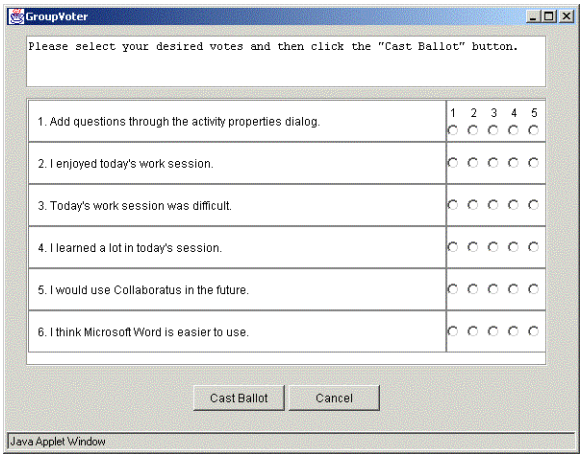

Figure 2. GroupVoter

Although Collaboratus does not meet every CW tool requirement, it represents a significant advance in CW tools. Some of the key requirements that are supported by Collaboratus include support of CW activities outside of drafting: Collaboratus has tools for brainstorming/brainwriting, outlining, and voting. Furthermore, Collaboratus supports group awareness and coordination by implementing a shared, group outline; annotations that refer to specific pieces of text and allow multiple levels of discussion within a given annotation; support of parallel partitioned writing on the same document by a virtually unlimited number of users; locking on the section level to help with coordination and conflict resolution; provision of different users roles and rights associated with those roles; providing a list of participants who are active in a given CW session; and version control capabilities that allow users to visually see the changes made to a particular document section over time. Figure 3 depicts the main group writing screen, while Figure 4 depicts an annotation being created.

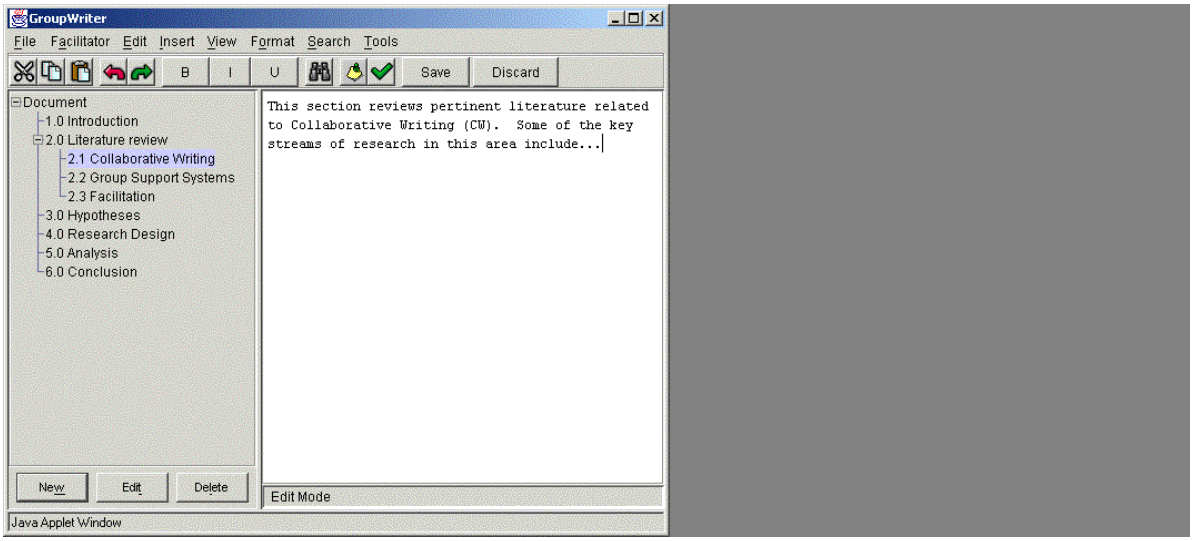

Figure 3. Example of GroupWriter Editing

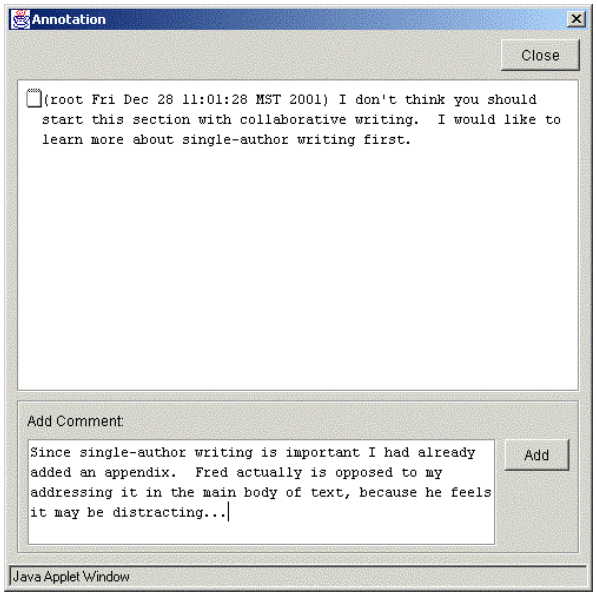

Figure 4. Example of an Annotation in GroupWriter

Another key set of requirements that Collaboratus supports is the ability to use virtually any type of client browser over the Internet. Furthermore, Collaboratus does not require any proprietary, client-side software installation. Because of its Java and web browser support, Collaboratus not only is easy to install but also is compatible with a virtually unlimited array of hardware and operating system platforms. Additionally, Collaboratus supports low bandwidth communications once the initial Java applets are downloaded. Some of the areas where Collaboratus does not fully meet the ideal requirements of CW tools, representing future research and development, include: negotiation support, communication about project scheduling and plans, in-depth social interactions, representation of who is working in the tree structure, and playback facilities to show a user what has occurred since their last log in. Despite these limitations, Collaboratus represents a significant advance in CW tools that warrants further research.

References

Adkins, M., Reinig, J. Q., Kruse, J., & Mittleman, D. (1999). *GSS collaboration in document development: Using GroupWriter to improve the process.* Paper presented at the Thirty-second Annual Hawaii International Conference on System Sciences, Hawaii.

Albrecht, C. C. (2000). *A programming framework supporting rapid application development of highly interactive, collaborative applications* (unpublished doctoral dissertation). Tucson, Arizona: University of Arizona.

Baecker, R. M., Nastos, D., Posner, I. R., & Mawby, K. L. (1993, April 24 - 29). *The user-centered iterative design of collaborative writing software.* Paper presented at the ACM Conference on Human factors in computing systems, Amsterdam The Netherlands.

Beck, E. (Ed.). (1993). *A Survey of Experiences of Collaborative Writing*. Berlin: Springer Verlag.

Couture, B., & Rymer, J. (Eds.). (1989). *Interactive Writing on the Job: Definitions and Implications of Collaboration*. Urbana, IL: NCTE and ABC.

Dubs, S., & Hayne, S. C. (1992, November 1-4). *Distributed facilitation: a concept whose time has come?* Paper presented at the Computer-supported cooperative work 1992, Toronto.

Ede, L., & Lunsford, A. (1990). *Singular texts/plural authors: Perspectives on collaborative writing*. Carbondale, IL: Southern Illinois University Press.

Ellis, C. A., Gibbs, S. J., & Rein, G. L. (1991). Groupware: Some Issues and Experiences. *Communications of the ACM, 34*(1), 39-58.

Erickson, T. D. (1989). Interfaces for cooperative work: An eclectic look at CSCW '88. *SIGCHI Bulletin, 21*(1), 56-64.

Fish, R. S., Kraut, R. E., Leland, M. D. P., & Cohen, M. (1988). *Quilt: a collaborative tool for cooperative writing.* Paper presented at the Conference on Office Information Systems (COIS).

Galegher, J., & Kraut, R. E. (1994). Computer-mediated communication for intellectual teamwork: An experiment in group writing. *Information Systems Research, 5*(2), 110-138.

Gere, A. R., & Abbott, R. D. (1985). Talking about writing: The language of writing groups. *Research in the Teaching of English, 19*(4), 362-385.

Gordon, M. D. (1980). A critical reassessment of inferred relations between multiple authorship, scientific collaboration, and the production of papers and their acceptance for publication. *Scientometrics, 2*, 193-201.

Greif, I., Seliger, R., & Weihl, W. (1986, 13-15 January). *Atomic data abstractions in a distributed collaborative editing system.* Paper presented at the Thirteenth Annual Symposium on Principles of Programming Languages, St. Petersburg, FL.

Horton, M., Rogers, P., Austin, L., & McCormick, M. (1991). Exploring the impact of face-to-face collaborative technology on group writing. *Journal of Management Information Systems, 8*(3), 27-48.

Kirby, A., & Rodden, T. (1995). *Contact: support for distributed cooperative writing.* Paper presented at the Fourth European Conference on Computer-supported Cooperative work, The Netherlands.

Koch, M. (1995). Design issues and model for a distributed multi-user editor. *Computer Supported Cooperative Work - An International Journal, 3*(3-4), 359-378.

Kraut, R. E., Galegher, J., & Egido, C. (1988). Relationships and tasks in scientific research collaboration. *Human-Computer Interaction, 3*, 31-58.

Lowry, P. B. (2002). *Improving distributed collaborative writing on the Internet using enhanced processes and a Java-based collaborative writing tool* (unpublished doctoral dissertation). Tucson, Arizona: University of Arizona.

Lowry, P. B., Albrecht, C. C., Nunamaker Jr., J. F., & Lee, J. D. (2002). Evolutionary development and research for an Internet-based Collaborative Writing tool to enhance eWriting in an eGovernment setting. *Decision Support Systems, Accepted for publication in 2002*.

McAlpine, K., & Golder, P. (1994). *A new architecture for a collaborative authoring system: Collaborwriter.* Paper presented at the Conference of Computer Supported Cooperative Work (CSCW).

Miles, V. C., McCarthy, J. C., Dix, A. J., Harrison, M. D., & Monk, A. F. (1993). Reviewing designs for a synchronous-asynchronous group editing environment. In M. Sharples (Ed.), *Computer Supported Collaborative Writing* (pp. 137-160): Springer-Verlag.

Moon, J. Y., & Sproull, L. (2000). Essence of distributed work: The case of the Linux Kernel. *First Monday, 5*(11).

Neuwirth, C. M., Kaufer, D. S., Chandhok, R., & Morris, J. H. (1990, 7-10 October). *Issues in the design of computer-support for co-authoring and commenting.* Paper presented at the Third Conference on Computer-Supported Cooperative Work (CSCW '90), Los Angeles, CA.

Neuwirth, C. M., Morris, J. H., Regli, S. H., Chandhok, R., & Wenger, G. C. (1998). *Envisioning communication: task-tailorable representations of communication in asynchronous work.* Paper presented at the Computer supported cooperative work, Seattle, WA.

Nunamaker Jr., J. F., Chen, M., & Purdin, T. D. M. (1991). Systems development in information systems research. *Journal of Management Information Systems, 7*(3), 89-106.

Olson, J. S., Olson, G. M., Storrosten, M., & Carter, M. (1992). *How a group-editor changes the character of a design meeting as well as its outcome.* Paper presented at the Conference on Computer-Supported Cooperative Work.

Posner, I. R., & Baecker, R. M. (1992). *How people write together.* Paper presented at the Twenty-fifth Hawaii International Conference on System Sciences, Kauai, Hawaii.

Schlichter, J., Koch, M., & Burger, M. (1997). Workspace awareness for distributed teams. In W. Conen (Ed.), *Lecture Notes on Computer Science*: Springer.