

Teaching Technical Writing for Computer Engineers Using the Web

Tara M. Madhyastha*

Abstract — *Technical communication is extremely important for engineering education (ABET) requirements, but limited resources make it challenging to offer writing-intensive classes in a growing engineering school. Use of online technologies can help writing-intensive classes to scale. We describe our experiences using a variety of Web-based functions to teach and administrate CMPE185, Technical Writing for Computer Engineers, at UCSC. Our major contribution is an online peer editing system for document markup that uses Adobe Acrobat® or Microsoft Word® for annotations. The peer editing software comes with a manual for both students and instructors and is available at <http://www.collage.soe.ucsc.edu>.*

Index Terms — *distance learning, document markup, peer editing, writing education*

1 INTRODUCTION

The University of California at Santa Cruz requires that all engineering students pass an upper-division course in technical writing. Students who take this course complete at least ten assignments (averaging approximately 50 pages) and an oral presentation within a ten-week quarter. Students must rewrite assignments until they pass them.

The growth of the engineering school and the lack of qualified instructors has been causing pressure to increase the class size to up to 80 students. Such enrollments pose a challenge to traditional classroom techniques for writing instruction, motivating us to find ways to approach class administration and writing education online.

The remainder of this paper is organized as follows. We describe the technical writing course and the traditional approach to its instruction in §2. We added online support for teaching and administration as described in §3. The COLLAGE PeerEdit system, developed as a result of ongoing course technology improvements, is surveyed in §4. We present related work in §5 and conclude with directions for future work in §6.

*Tara Madhyastha, Department of Computer Engineering, University of California Santa Cruz, 1156 High Street, Santa Cruz, CA 95064 tara@soe.ucsc.edu

2 TECHNICAL WRITING FOR COMPUTER ENGINEERS

Technical Writing for Computer Engineers was created in 1989 by Kevin Karplus and Dan Scripture as an engineering course requirement. Students who pass the course should be capable of clear, concise writing and presentation on technical subjects. To this end, it is a writing-intensive class with ten written assignments and an oral presentation during a ten-week quarter.

The first assignment is to write a resume and cover letter for an actual job posting, which illustrates to students how to write to a specific audience. Subsequent assignments include writing a hiring memo, describing how to implement a more efficient sorting algorithm, commenting code, writing an HTML manual for a naive user, a survey article, and a technical final project of the student's choice [8].

This course has been evolving over the last thirteen years. It has been a rule of thumb that 20 students require 20 hours of grading and office hours help. In the beginning, the class was effectively team-taught by a writing professor and a computer engineering professor, allowing the instructors to merge their skills and while handling up to 45 students. In recent years, it has been taught entirely by one engineering professor, with teaching assistants to handle the grading workload.

There is a large emphasis on in-class peer editing; students bring their assignments and work with each other to review them. All assignments are available electronically in the CMPE185 workbook [8] but are submitted and graded on paper hardcopy.

3 SUPPORT FOR WEB-BASED INSTRUCTION

In Winter 2000 I began making changes to CMPE185 to accommodate growing enrollments. During that quarter there were 40 students enrolled in CMPE185; in Winter 2001, there were 79, and in Fall 2001, there were 70. This growth required some significant changes in course management. At 20 students per instructor or teaching assistant (TA), the 2001 offering required 3 TAs, which had the

potential to cause a serious consistency problem in grading. Peer editing becomes unmanageable in such large class sizes. We were interested in understanding how technology could provide solutions to these problems.

In the Winter and Fall 2001 quarters, we used WebCT [3], a Web-based course authoring software package, to administer the course. This software enabled many of the improvements described below.

ONLINE SUBMISSION AND GRADING

The first change, implemented in Fall 2000, was online submission and grading using Adobe PDF and Adobe Acrobat. Adobe PDF is a universal file format that preserves all the fonts, formatting, graphics, and color of any source document, regardless of the application and platform used to create it [4]. It is an open standard and reader software is freely available. Adobe Acrobat is a product that allows the creation and markup of PDF documents.

Students were required to submit their assignments online in PDF or PostScript format (not all lab computers were equipped with support to generate PDF until Fall 2001) before midnight on the evening assignments were due. Students could view their submitted assignments through the same online interface.

The instructor and TAs then used Acrobat to convert the documents to PDF, if necessary, and to make comments on the submitted assignments, which the students could view in the same way. By having both regular meetings to discuss grading criteria for each assignment, and access to other grader's comments, we were able to keep grading style and format consistent. We each had access to the graded assignments at any time, without photocopying them or relying upon the students to hand them back to the instructor for review, techniques used in previous quarters. Grades for assignments were entered manually into WebCT, allowing students to verify that they were recorded correctly.

Of the 72 respondents to the exit survey taken in Winter 2001, 80% preferred online assignments and grading to assignments and grading in hard copy. More specifically, 66% reported liking being able to submit assignments in PDF format, and 75% reported liking the online grading. Ninety-nine percent of the students reported liking the WebCT feature that allows them to check their grades online.

SCHEDULING OFFICE HOURS

The editing and rewriting process is extremely important in this technical writing course, so the instructor, teaching assistants, and readers must schedule many office hours. In the past, handling students during office hours and tutor reading hours has been difficult; often as

many as ten people would wait outside the instructor's office for hours, waiting their turn for help. Sign-up sheets were frequently ignored or lost.

WebCT provides a calendar on which students can make public entries; we used this to both schedule instructor, TA and tutor office hours and to allow students to schedule time with us. Our goal was better utilization of office hours, less wasted time for students and TAs, and easy statistics for number of times that students go for help. In the exit survey, 71% of the students said they liked the online office hour signup.

Although the advantages of this signup system are obvious, we encountered some problems in practice. Some students signed up for slots but didn't show up, while others didn't sign up but dropped in for help anyway. According to our exit survey, 17% of the students reported signing up and not showing up, while 75% said they always showed up to appointments. The reason most frequently given for not showing up to an appointment was "something came up"

ONLINE MARKUP-BASED PEER REVIEW

Peer review, where students exchange assignments and review each other's text, is a time-honored technique in writing education. Unfortunately, as the class size increases, there are too many students to manage for peer editing to be effective. Furthermore, peer editing takes up valuable class time, it is difficult to assess the quality of student's peer edits, and students are reluctant to criticize each other face-to-face.

Our approach has been to support markup-based peer review, where students use the document markup and editing capabilities provided by Adobe Acrobat and Microsoft Word to access and edit other students' documents on the Web. Figure 1 shows an example of a hiring memo (with identifying names changed) that has been edited by several students and graded by the instructor. Students can use a variety of tools to "draw on" the text (e.g., pen, highlighter) and can strike out text. They can distinguish their comments by selecting identifying shapes and color codes. In Figure 1, some comments have been opened, making their contents visible, and the rest are closed.

The other main category of peer review system is "calibrated peer review," where students answer questions about another text and rank it according to various attributes. Rankings may be combined to obtain a score for the paper, similar to conference paper review. Unfortunately, it is difficult to edit text this way. In document markup-based peer review, the instructor is not bound to this infrastructure, and can specify that the students make comments at any granularity. The drawback, however, is that it is more difficult for the instructor to use peer editing comments to grade or evaluate a student paper.

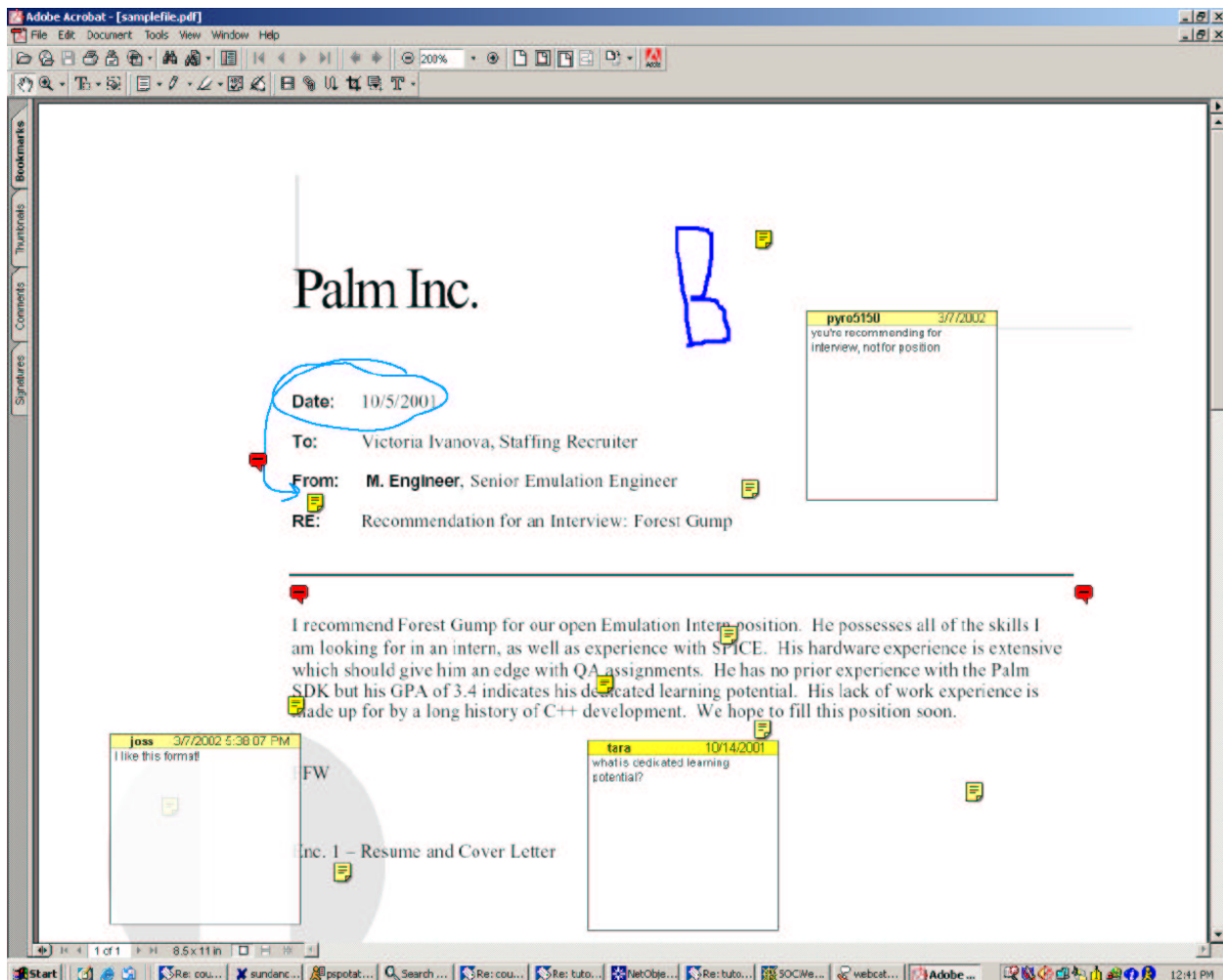


Figure 1: Example of markup-based peer review.

INITIAL EXPERIMENT: WINTER 2001

We extended the concept of online grading to online peer editing. In Winter 2001, we created 20 online private discussion groups with five students each, using alphabetical order to assign students to groups. Normally students submit an assignment, we grade it, and they can rewrite it on the basis of our comments. In this experiment, we asked the students to post their assignments to their private discussion group on the due date, and informed them that they could rewrite the assignment only if they did a good job of peer editing the documents of two others in their group. We posted specific grading criteria on the Web.

Students were given a choice of either writing their comments and posting them, or using Adobe Acrobat to markup the text. The peer edits were outstanding compared to editing accomplished in class. In the exit survey, 65% of students said they preferred online peer editing to in-class peer editing. We noticed that the quality of

the online peer edits was excellent compared to in-class peer edits, and hypothesized that the anonymity of online peer editing might contribute to this phenomenon. As suspected, 39% of students reported that they agreed with the statement "I feel more comfortable editing the document of someone I don't know than someone I do," and 47% agreed with the statement "I feel more comfortable editing the document of someone I don't have to see than someone I am sitting next to."

We noticed that students within each group conformed to the style of the comments (document markup or text) of the first posting of the group, and in general, mimicked the style of the grading that had been done so far (down to assigning actual grades, including huge NP's (for No Pass) scrawled across the top of the first page).

DEPLOYED SYSTEM: FALL 2001

The Winter 2001 experiment was promising enough to

lead us to build and deploy a more comprehensive peer-editing system in the Fall 2001 course offering, called COLLAGE (COLLaborative Approach to Global Education) PeerEdit. Our approach was to use the document editing features of Adobe Acrobat and Microsoft Word as a foundation for document markup, and wrap these with the authentication, authorization, and student tracking services necessary to use these commodity tools in a courseware environment. This approach has two main advantages: first, students continually benefit from the new features available in commodity tools, and second, they do not need to learn to use a new tool. From the developer standpoint, we can exploit the features offered by this software and do not need to reinvent them.

In Fall 2002, we obtained Adobe Acrobat licenses for the computer labs and required students to submit their assignments in PDF format, and to do peer edits using Acrobat. We chose Acrobat (rather than Microsoft Word) because of its more flexible suite of markup options. Students can insert post-it note style comments, “write” on the document with a pen or highlighter, and strike out text.

Students were required to submit two drafts of each assignment (they would receive a failing grade for the assignment if they did not submit the first draft). Two peer edits of the first draft of each assignment were required; however, they were allowed to do up to four for extra credit.

Although we did not do an exit survey in Fall 2001, students were anecdotally happy with the peer editing. The quality of the edits was consistently quite high. Many comments gave insight into what information made it through to the class (e.g., students would repeat what I said, and sometimes get it wrong). Some comments were socially supportive: for example, many students wrestled with Microsoft Excel to generate a graph for an assignment on presenting data visually, and expressed sympathy and suggestions for others they saw fighting the same problems.

4 COLLAGE PEEREDIT

Space constraints prevent complete description of all the features of the PeerEdit software, but those interested may refer to the user manual [5]. Instead, we describe the design of the system and summarize the supported functions.

DESIGN

Figure 2 shows an overview of the system components. Major functions are divided into blocks; currently students authenticate and join a class to be authorized to engage in peer editing. Instructors may also authenticate and

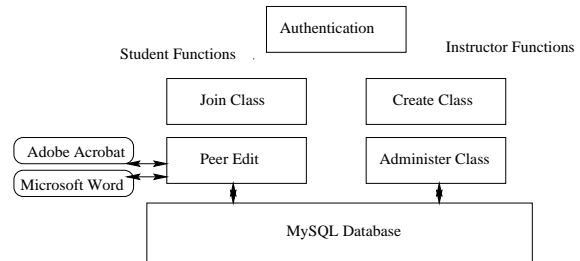


Figure 2: COLLAGE PeerEdit system components.

create and administer classes. These functions are implemented as Perl CGI scripts. A MySQL database stores student data, instructor data, class information, and peer editing state information. This infrastructure permits the use of external software packages (e.g., Adobe Acrobat and Microsoft Word) in the context of a classroom setting.

GENERAL FEATURES

Course authoring systems such as WebCT manage course creation, student authentication, student authorization, and a variety of other functions. Unfortunately, the lack of standards in this domain make it impossible to share this functionality when designing a new courseware component. This is the concern being addressed by the Open Knowledge Initiative [1]. Therefore, PeerEdit provides its own way for instructors to create accounts and classes, and for students to create accounts and join classes. Access control to classes is done via a passphrase that the instructor provides to the members of the class. This replication of functionality is not ideal, and we will attempt to conform to standards for student information and class lists as they evolve.

INSTRUCTOR FEATURES

Once they have created their classes, instructors can create assignments in each class. Each assignment has two deadlines: a submission deadline and a peer edit deadline. Students must submit their assignment by the submission deadline, and then they may begin doing peer edits. An instructor can specify that there is no peer editing by making the peer editing deadline the same as the submission deadline. Each draft is generally its own assignment, and the last draft will have no peer edits. Not every draft needs to be graded by the instructor.

Instructors may examine a table of peer editing activity where rows correspond to students and columns are assignments. In this way, they can obtain a count of edits per student and view the quality of each edit.

To grade an assignment, instructors download a gzipped tar file of the assignments, both with peer edits

and without. Thus, instructors who prefer to grade on paper may print the unedited versions (comments in PDF may obscure text). However, if assignments are graded electronically, they may be uploaded so that students can view the grader comments together with the peer edits.

STUDENT FEATURES

Students may perform any of four peer editing functions. They can view the list of assignments and deadlines, submit an assignment, view submitted assignments and peer editing status, or edit other students' work. Figure 3 shows the table of assignments from Fall 2001. Assignments are color-coded to highlight upcoming deadlines. Students may not submit late assignments, and they must be in the format specified by the instructor (PDF or Word). At any time, they can see how many people have edited their assignment, so they can view comments once the edits are complete. They may also view the assignments that they have edited, to see what other students have to say, and if assignments are graded electronically, to view grader comments. This allows instructors to give feedback to the peer editors about their comments.

When a student elects to edit another student's work, he or she is presented with a link to another student's assignment. There is currently only one algorithm for assigning peer edits: assignments are sorted first in order of submission, second by number of completed peer edits. A student who requests an assignment to edit receives the first assignment in the sorted list that he or she did not author and has not previously edited. Once the student (editor) has "checked out" an assignment for peer editing, it cannot be edited by anyone else until the editor has checked it back in. This simple locking mechanism prevents students from overwriting each other's edits. Each student may do up to four edits for each assignments. If there is nothing available to edit (e.g., there are only three students in the class and one tries to do three edits, or all assignments are currently being edited) the student will be informed that there is nothing to edit, and they might try again later.

When a student checks back in an assignment, it is compared to the original submission to ensure that the text is identical. This prevents students from accidentally overwriting an assignment. As an additional safeguard, original copies of unedited assignments are saved and are accessible only to instructors.

5 RELATED WORK

Many peer editing systems are examples of calibrated peer review, where instructors specify questions and students provide answers to them. Much like conference review, scores along a particular scale can be calibrated to provide summary feedback as well as detailed comments.

Example of such systems include Calibrated Peer Review (CPR) from UCLA [6] and Turnitin.com [2].

Online peer editing is a specialized application of collaborative authoring, which is supported most commonly by the Web-based Distributed Authoring and Versioning (WebDAV) protocol [7]. This protocol specifies extensions to HTTP to allow online collaborative writing. Adobe Acrobat 5.0 supports WebDAV to allow users to simultaneously mark up documents on the Web. Unfortunately, support for collaborative authoring is not enough to administrate peer editing in a large class; one needs very fine control over student access, a method for allocating assignments to students, and a way to track student activities.

6 CONCLUSIONS AND FUTURE WORK

We have identified a variety of online technologies to help to teach and administer a large technical writing class, including online submission and grading, web-based office hour scheduling, and markup-based peer review. We describe the peer review system, which is documented and available at www.collage.soe.ucsc.edu.

COLLAGE PeerEdit is an ongoing project, and is under constant development. We are exploring ways of supporting greater editing concurrency, perhaps exploiting WebDAV support in Adobe Acrobat and Microsoft Word. When this software was developed, it was unclear that students would have access to WebDAV-enabled software. We are also expanding the algorithms for assigning students to documents to edit. Finally, we are integrating this software with plagiarism checking software.

7 ACKNOWLEDGMENTS

I am grateful for the patience of three generations of CMPE 185 students who tested this software. In particular, Jael Aumack devoted many hours in Fall 2001 to enhancing the software and documenting the system. This work was supported in part by the National Science Foundation under grants NSF CCR-0093051.

References

- [1] Open knowledge initiative. <http://web.mit.edu/oki>, Mar. 2002.
- [2] Turnitin.com. www.turnitin.com, Accessed June 2002.
- [3] Webct.com. <http://www.webct.com>, Mar. 2002.

**CMPE185: Technical Writing for Computer Engineers --
Fall 2001**

Assignment Due Dates

This is the one true source of due dates. Accept no substitutes. All assignments are due on midnight of the due date.

Assignment	Due Date	Peer Edits Close	Format
Resume and Cover Letter	September 29, 2001	October 1, 2001	PDF
Hiring Memo	October 5, 2001	October 8, 2001	PDF
Describing an Algorithm	October 12, 2001	October 15, 2001	PDF
In-Program Documentation	October 16, 2001	No Peer Edit	PDF
REWRITE: Hiring Memo	October 18, 2001	No Peer Edit	PDF
Naive User Documentation	October 26, 2001	October 29, 2001	PDF
Survey Article	October 30, 2001	November 2, 2001	PDF
REWRITE: Describing an Algorithm	November 1, 2001	November 3, 2001	PDF
Term Project Proposal Memo	November 3, 2001	November 6, 2001	PDF
REWRITE Naive User Documentation	November 8, 2001	No Peer Edit	PDF
REWRITE Survey Article	November 12, 2001	No Peer Edit	PDF
Term Project	November 25, 2001	November 28, 2001	PDF
REWRITE Term Project	December 3, 2001	No Peer Edit	PDF

Figure 3: Table of assignment due dates.

- [4] Adobe. Adobe pdf. umber engineers and computer scientists. <http://www.adobe.com/products/acrobat/adobepdf.html>, Mar. 2002. <http://www.cse.ucsc.edu/~karplus/185/f00/reader.ps>, Sept. 2000.
- [5] J. M. Aumack. Collage peer-editing software manual. <http://www.collage.soe.ucsc.edu/peerEdit/doc/CollagePeerEditUserManual.pdf>, Jan. 2002.
- [6] O. L. Chapman. Calibrated peer review. <http://cpr.molsci.ucla.edu/>, March 2002.
- [7] J. E. James Whitehead. Design spaces for link and structure versioning. In *Proceedings of Hypertext '01, the 12th ACM Conference on Hypertext and Hypermedia*, Arhus, Denmark, Aug. 2001.
- [8] K. Karplus and D. Scripture. Workbook for cmpe 185: Technical writing for com-