Writing, Reviewing, and Rewriting in Upper-Level Computer Science Courses

Bryant A. Julstrom
Department of Computer Science
St. Cloud State University
julstrom@stcloudstate.edu

Abstract

Computing professionals spend a large part of their time writing a wide variety of documents, so it is essential that they write well. As innumerable authorities on writing have pointed out, careful revision is the essential path to good writing. Computer scientists often submit their writing to refereed forums like conferences or journals, where it is reviewed, and they review the work of other authors. Computer science students gain experience in writing, revising, reviewing, and being reviewed when projects in upper-level computing courses include papers that are reviewed and revised as for a conference.

This conference model has been used in a variety of upper-level computing courses. Students review each others' papers; they undertake revisions in response to those reviews and produce improved papers; and they recognize the benefits of the process.

Introduction

Though the uninformed may still harbor the illusion that computing consists mainly, if not exclusively, of coding, we know that computer scientists produce far more natural language than code (*e.g.*, Sullivan [13]). They write problem descriptions, program designs, program and user documentation, grant proposals, reports, papers, articles, and more. It is therefore essential that computer scientists write well. Modern computing curricula emphasize good writing, as do the ACM/IEEE curriculum guidelines [6, p.42] and the computer science accreditation standards of the Accreditation Board for Engineering and Technology (ABET) [1, p.4].

Effective rewriting is the key to good writing. The first draft of a document is rarely acceptable; careful editing and rewriting produce clear, concise, readable prose. As Higham points out, "All writing benefits from revision." [5, p.94]

Computing professionals are likely, occasionally or regularly, to find themselves writing for submission to refereed forums--conferences or journals--and to be called on to review such submissions. After submitting a paper to a conference or journal, its authors receive, along with notification of its acceptance or rejection, reviews of the paper. If the paper has been accepted, the authors are obliged to consider the reviewers' observations as they prepare the paper's final version. CS coursework should expose students to the processes of submitting and reviewing papers. As Pesante [12] points out, "Students … need an initiation into the computer science and software communities." Reviewing and being reviewed are among the activities of those communities.

To encourage and enable students to edit and rewrite, and to acquaint them with the processes of reviewing and being reviewed, I have instituted research projects in senior-level courses whose writing component is modeled on submitting papers to a conference. For these assignments, students propose and carry out large projects and describe those projects in conference-style papers. Other members of the class and the instructor review the papers, and the students consider these reviews in preparing their final drafts. They also make oral presentations based on their papers.

This mechanism has been used recently in 400-level courses in Neural Networks, Artificial Intelligence, Evolutionary Computation, and (currently) Expert Systems in the Department of Computer Science at St. Cloud State University. Students report that, after the opportunity to rewrite in response to readers' reactions to their papers, they feel more confident in the quality of their work, and the papers are indeed improved.

The following sections of this paper consider the process of editing and rewriting in the context of computer science, describe the reviewing process, present the conference model of paper preparation, and summarize the results of following this model, both effects on writing quality and students' reactions.

Editing and Rewriting

Authorities on writing are unanimous in asserting the importance of careful, thorough revision in producing clear, concise, readable prose. Kay [9], for example, points out that "Writing is an iterative process of enhancement, revision, and polish." Many authors suggest writing the first draft as quickly and freely as possible, so as to get on paper the raw material for revision. As Zobel [16, p.8] observes, "Many writers find it helpful to write freely … so that they can concentrate on presenting a smooth flow of ideas," and, more forcefully, "If you tend to get stuck, just write anything, no matter how awful."

However a first draft is constructed, it is rarely acceptable as finished work. More often than not, it is cluttered with errors of organization, style, and even grammar. Nonetheless, this draft provides the basis for successive steps of rewriting and editing, through which the author incrementally improves and polishes the document and arrives at its final form. In general, only careful revising will produce a readable, informative product. To quote Zobel once more, "The best writing is the result of frequent, thorough revision."

Several authors note, with Anewalt [2], "the similarity between the writing process and the software design process," and invoke that similarity both to improve the quality of students' writing and to "make computer science students feel more connected to the writing process." The similarity lies, of course, in the processes of top-down design and stepwise refinement. Programmers elaborate program designs, implement them, and flesh out their details. At each step, they test and correct previous work. Similarly, writers (ideally) write from a plan, working through a series of drafts, each more polished than the last. Taylor and Paine [15], Kay [9], and others have made this connection explicit. (Conversely, Ladd [10] described a writing-inspired multi-draft model of programming in CS1 and CS2. Null *et al*. [11] also applied paper-rewriting models to the construction of software.)

More generally, several authors (*e.g.*, Jackowitz *et al*. [7], Hafen [4], and Kay [9]) describe multi-step writing projects in computer science courses that involve drafts and revisions. The conference model described below is one of these.

Reviewing

Reviewing is the careful and critical reading of others' work and the writing of observations about and suggestions for that work. It is a regular obligation for many computing professionals, especially academics. It is a crucial step in the process of seeing an article published. And it encourages and requires careful, close reading. For all these reasons, computer science students should be introduced to the process of reviewing, both writing reviews and being reviewed themselves. Upper-level writing projects offer an opportunity for this introduction: students can review each others' work.

This process is called peer reviewing, and many authors have described writing projects in computer science that include peer review. Hafen [4], for example, described peer review of term papers in a senior-level database course. Students not only wrote comments but also participated in workshop-style sessions to discuss the papers. Gehringer [3] presented a web-mediated peer review mechanism. Sullivan [14] described the use of peer review to evaluate presentations and software. Kaczmarczyk [8] used peer review in a course devoted specifically to technical writing for computer science.

Peer review has the additional benefit of lending credibility to the instructor's criticisms. Students sometimes claim that if an idea is on paper, they should get credit for it, regardless of the artlessness of its presentation. When other students suggest that the presentation is poor, they are often more likely to take such observations seriously.

The Conference Model

In imitation of the submission of papers to a conference, projects in upper-level courses consist of these steps: proposal, papers, reviews, and revised papers. This structure, of course, is not original. Hafen [4], for example, describes writing assignments in a database course that include "several stages: topic selection, first draft, peer critiques, and final paper."

Projects begin with one-page proposals in which the students outline what they plan to do and how. Proposals give the instructor the opportunity to intervene early to widen investigations that are too narrow, narrow those that are too broad (more frequently), and generally avoid likely difficulties. Proposals have the added benefit of discouraging procrastination.

The format of the papers is precisely specified. The title and author(s) occupy one column that reaches across the page, but the body of the papers is in two columns of 10-point type. There must be an abstract, independent of the text that follows it. Tables and figures are numbered, with captions. Table captions precede tables, while figure captions follow figures. The name-date style is used for references: "… (Jones, 1993)" or "Jones (1993) reported that … ."

Most important, papers may be no more than five pages long. In the two-column format, this is ample space, but it imposes some discipline and warns against windiness. Students often express surprise at the requirement and (later) at the thought and care required to adhere to it.

The students hand in multiple copies of their papers for reviewing. The instructor gets one copy of every paper, and the remaining copies are randomly redistributed so that every students gets two papers to review and every paper gets at least two reviews in addition to the instructors'. In their comments, students are encouraged to be both

specific and helpful. “Say what?” is not a useful comment, but “I don’t understand the quantity *r* in the evaluation function. An example?” is.

Figure 1 shows the reviewing form; it is adapted from one used by the ACM Symposium on Applied Computing. In general, reviewing is not blind, a departure from most conferences’ practice, but in small, upper-level courses, the students generally know who is working on what, so blind reviewing is impossible.

```
Author(s): _____________________________________________________________

Title: _________________________________________________________________

================================================================
EVALUATION:
                                    tend to reject tend to accept
                                    <-------------|------------->
Technical Content and Accuracy      1  2  3  4  5  6  7  8  9  10
Originality                         1  2  3  4  5  6  7  8  9  10
Replicability                       1  2  3  4  5  6  7  8  9  10
Significance                        1  2  3  4  5  6  7  8  9  10
Title/Introduction/Conclusion       1  2  3  4  5  6  7  8  9  10
Organization                        1  2  3  4  5  6  7  8  9  10
Style and Clarity                   1  2  3  4  5  6  7  8  9  10
Adequacy of References              1  2  3  4  5  6  7  8  9  10
Adherence to Standards              1  2  3  4  5  6  7  8  9  10

OVERALL EVALUATION:                 1  2  3  4  5  6  7  8  9  10
----------------------------------------------------------------
COMMENTS:
```

Figure 1: The reviewing form used in the conference-model projects.

The students have a few days to review the papers, then the reviews are distributed to the papers’ authors. Based on the reviews, the students revise their papers and turn in the revised versions. It is on the revised versions that their evaluations are based.

The projects conclude with oral presentations ten to fifteen minutes long, and every student receives a copy of the course’s *Proceedings*: all the final versions with a table of contents, covered and bound.

Results

So far, the conference model has been a success. First, students participate willingly in the process. Though their reviews are not tallied or graded, students feel an obligation to each other and generally get the reviews done promptly; they are eager to see readers’ reactions to their own work; and they take those reactions seriously as they revise their papers.

Second, while a few students use the opportunity to revise only to correct the errors of spelling and punctuation that the reviewers found and ignore larger issues of content, organization, and presentation, most revise conscientiously and carefully in response to the reviews.

Third, students like the process. Evaluations in these courses include an open-ended question about the steps of reviewing and revising in projects. Responses to this question have been overwhelmingly positive. Comments have included, “A great experience in writing a paper,” “The more feedback, the better,” and “Having a second go at the paper was wonderful.”

There is, of course, a down side. Some students use the revision period not to revise a carefully constructed paper but to complete one. Their first submissions are hasty, sketchy, often incomplete. A conference would reject such papers, but these are courses.

The students’ reviews are sometimes superficial. The one critical observation from the course evaluations mentioned this, observing that the instructors’ comments were more helpful than the other students’.

And the conference model creates more work for the instructor, who must now read and evaluate every paper twice rather than just once. In a small class, this is no problem, but with a large group it can be daunting.

Conclusion

Computer scientists write a wide variety of documents so it is essential that they write well; revision is the essential step toward this end. Computer scientists review submissions to conferences and journals, and their submissions are reviewed in turn. Students gain experience in all these tasks when projects in upper-level courses include papers that are submitted, reviewed, and revised as for a conference. This conference model, similar to writing projects that many authors have described, has been used successfully in a variety of upper-level courses. In spite of some malingering, students in general participate enthusiastically, produce both useful reviews and improved papers, and find the process helpful.

The model could be extended in a variety of ways. No doubt the final papers would be improved by earlier feedback, more rounds of reviewing, and workshop-style sessions on the papers’ mechanics. Similarly, students could evaluate the reviews of their work. However, these projects take place in courses devoted to particular topics in computing. That material must remain their focus while we use the conference model to incrementally improve students’ writing.

References

1. Accreditation Board for Engineering and Technology (ABET) (2003). Criteria for accrediting computing programs. Baltimore, MD, 2003. http://www.abet.org/criteria.html.

2. Anewalt, Karen (2003). A professional practice component in writing: A simple way to enhance an existing course. *The Journal of Computing in Small Colleges*, V.18, n.3, pp.155-165.

3. Gehringer, Edward F. (2001). Electronic peer review and peer grading in computer-science courses. *ACM SIGCSE Bulletin*, V.33, n.1, pp.139—143.

4. Hafen, Marguerite (1994). Developing writing skills in computer science students. *ACM SIGCSE Bulletin*, V.26, n.1, pp.268-270.

5. Higham, Nicholas J. (1993). *Handbook of Writing for the Mathematical Sciences*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

6. IEEE Computer Society, Association for Computing Machinery (ACM) (2001). Computing curricula 2001 computer science. http://www.computer.org/education/cc2001/final.

7. Jackowitz, Paul M., Richard M. Plishka, and James R. Sidbury (1990). Teaching writing and research skills in the computer science curriculum. *ACM SIGCSE Bulletin*, V.22, n.1, pp.212-215.

8. Kaczmarczyk, Lisa C. (2003). A technical writing class for computer science majors: Measuring student perceptions of learning. *ACM SIGCSE Bulletin*, V.35, n.1, pp.341-345.

9. Kay, David G. (1998). Computer scientists can teach writing: An upper division course for computer science majors. *ACM SIGCSE Bulletin*, V.30, n.1, 1998, pp.117-120.

10. Ladd, Brian C. (2003). It's all writing: Experience using rewriting to learn in introductory computer science. *The Journal of Computing in Small Colleges*, V.18, n.5, pp.57—64.

11. Null, Linda, Mike Ciaraldi, Liz Adams, Ursula Wolfe, and Max Hailperin (2002). Rewrite cycles in CS courses: Experience reports. *ACM SIGCSE Bulletin*, V.34, n.1, pp.249—250.

12. Pesante, Linda Hutz (1991). Integrating writing into computer science courses. *ACM SIGCSE Bulletin*, V.23, n.1, pp.205-209.

13. Sullivan, Sarah L. (1988). How much time do software professionals spend communicating? *ACM SIGCPR Computer Personnel*, V.11, n.4, pp.2-5.

14. Sullivan, Sarah L. (1994). Reciprocal peer reviews. *ACM SIGCSE Bulletin*, V.26, n.1, pp.314-318.

15. Taylor, Harriet G. and Katherine M. Paine (1993). An inter-disciplinary approach to the development of writing skills in computer science students. *Proceedings of the Twenty-Fourth SIGCSE Technical Symposium on Computer Science Education*. Indianapolis, IN, pp.274—278.

16. Zobel, Justin (1997). *Writing for Computer Science: The Art of Effective Communication*. Singapore: Springer-Verlag.