WRITING FOR COMPUTER SCIENCE: A TAXONOMY OF WRITING TASKS AND GENERAL ADVICE*

Robert F. Dugan Jr.
Department of Computer Science
Stonehill College
Easton, Massachusetts 02357
bdugan@stonehill.edu

Virginia G. Polanski
The Writing Program
Stonehill College
Easton, Massachusetts 02357
vpolanski@stonehill.edu

ABSTRACT

Computer science graduates lack written communication skills crucial to success in the workplace. Professional and academic organizations including ACM, IEEE, ABET, CSAB, and NACE have stressed the importance of teaching computer science undergraduates to write for years, yet the writing problem persists. In this paper we provide guidance to computer science instructors who want student writing skills to improve. First, we organize prior work on writing for computer science into a goal-oriented taxonomy of writing tasks. Each task includes a clear, concise, and detailed model that can be used as the framework for a student writing assignment. Second, we provide general advice for incorporating writing into any computer science course. Finally, we discuss the application of our taxonomy and advice to writing tasks in several computer science courses.

INTRODUCTION

Despite the importance of writing stressed by organizations like ACM, CSAB, and ABET, computer science graduates lack written communication skills crucial to success in the workplace. In this paper we provide guidance to computer science instructors who want student writing skills to improve. We organize prior work on writing for computer science into a goal-oriented taxonomy of writing tasks. Each task includes a clear, concise, and detailed model that can be used as the framework for a student writing assignment. We provide general advice for incorporating writing into any computer

science course. Finally, we discuss the application of our taxonomy and advice to writing tasks in several computer science courses.

Our paper begins with two narratives. We believe that our backgrounds (over a decade programming in industry and a half-century of teaching writing) provide a unique perspective on the problem of writing for computer science.

Computer Science Professor Narrative

As a computer science undergraduate at an engineering school, I had a pretty dim view of writing. One of my friends reflected this common view with "I chose computer science so that I'd never have to take an English course again." I thought that most of my professional career would be spent creating interesting software and, of course, making lots of money. This view changed during a summer internship at IBM when I was asked to write the user manual for some software I had developed. It had taken me a month to write the software, and I figured it would take me a week to write the manual. A week later, I submitted my first, and I assumed, final draft to my supervisor and went back to programming. Several days later, the draft was returned, "dripping" with red corrections. Another week went by as I made corrections, and submitted my "new final draft." Several days later that draft was returned with more red ink. Several months later, both my supervisor and I were tired, but happy with the manual. I had spent twice as much time writing as programming in that internship. My hard work paid off, however, when a users group voted the manual "most likely to take to a desert island."

Despite editorial challenges in my manual, upon graduation I was hired by the same supervisor and worked at IBM for almost ten years. As my career progressed I began to notice that the people who advanced quickly were both competent programmers and excellent written and verbal communicators. I found myself writing more user, requirements, functional specification, and design documents. At my supervisor's request, I delivered presentations, and produced written status reports, regularly. Occasionally I actually got to write software!

Writing was not always a smooth process. On a number of projects my team was asked to create a particular type of document, (for example requirements), without clear guidelines for the document's structure. When development ground to a halt because of a technical issue, the issue would often be resolved in a furious email exchange between team members. And it was not always resolved in favor of the best technical resolution, but sometimes in favor of the more effective communicator. When I worked for a manager who did not write well, I agonized over how my yearly evaluation would be interpreted. When I was promoted I found myself in the uncomfortable role of document reviewer. I often had to restrain myself from completely rewriting the document I was reviewing.

Years later, I accepted a position as assistant professor of computer science at a small liberal arts college. My career in industry had taught me that computer scientists do a significant amount of writing, but many lack writing skills. I wanted to improve my students' writing, but I needed help. Experience had made me a better writer despite an engineering background, but I wasn't sure I could help students develop these skills. I

approached a colleague who had been teaching writing for about fifty years, and together we hatched the idea of co-teaching a software engineering course.

Writing Professor Narrative

I knew I could work with computer science students to help them improve their writing skills. I had worked with engineers, scientists, and other professionals and business people in writing classes on campus and on consulting jobs at firms and companies. I assumed the students would be motivated because they were close to graduation and jobs in industry. Some or all of them had already done or were involved in internships.

However, I also assumed that these students would question my competency to write in their field. Being considered incompetent outside of my own discipline went with the territory. It was a reaction to the myth that only an expert in another discipline could write in that discipline. But I had been using my rhetorical and grammatical skills to analyze models and to determine formats to meet needs in many disciplines and fields. I had helped traditional students, business and professional people, and other academics improve their writing and sometimes to get it published.

My technical writing students had prepared manuals for real clients drawing information from the clients themselves, model documents, textbooks, and outside consultants. My organic chemistry students had used the *ACS Style Guide: A Manual for Authors and Editors* with guidelines for posters, letters to the editor, press releases, oral presentations, ACS books, journal and magazine articles, scientific research articles, abstracts, and documentation. These students had also analyzed articles from professional journals and used them as models for preparing articles about their own research projects.

With the success of these experiences, I was excited about working in another new and challenging situation when asked to teach a writing component for a computer science course. I just wanted my computer science colleague to provide the official guidelines for preparing workable documents and publishable articles in the computer science field. When I found out that the computer science field didn't have such guidelines, I asked for professional models for the requirements and design documents and the user manual that I would be expected to help students write. When I found out that the available models were inconsistent, I knew my challenge would be greater than I had expected.

PRIOR WORK

Researchers have been studying the problem of writing for computer science for several decades. Common sources of research include *ACM SIGSE* conferences, *ACM Journal of Computing in Small Colleges*, *IEEE Frontiers in Education* conferences, and the *IEEE Transactions on Professional Communication*. As recently as 2003, work was presented at the *CCSCNE* conference on the use of portfolios and rewriting in early programming courses [Ladd2003].

Generally speaking, prior research has attacked the writing problem with either a broad or focused application of writing tasks. Broad solutions apply a small number of writing tasks across the entire computer science curriculum (for example, using the agile development methodology in any course to improve student writing [Kussmaul2005]). Focused solutions apply a single writing task to a specific set of courses (for example, using diary/journal writing in introductory computer science courses [Fekete2000]).

The small-college-computer-science instructor generally teaches a variety of courses. Each course has the potential to address the writing problem in a different manner. As a result, the instructor is responsible for assigning a large variety of writing tasks. We believe that instructors would benefit from guidance for assigning these writing tasks. No single solution from the prior research can provide this guidance because each solution applies to a small number of writing tasks or courses. Therefore, we decided it is time to take a step back and organize the research and praxis into a comprehensive taxonomy of writing tasks.

TAXONOMY OF WRITING TASKS

Our taxonomy is divided according to three main goals: writing for learning, writing for academic communication, and writing for industrial communication. Writing tasks within each goal are grouped into goal-oriented categories of similar tasks. Each task includes a description and citation. The citation provides a clear, concise, and detailed model for the writing task. Instructors can use this model when assigning the task to students. The taxonomy is related to Orr's work, but focuses on student writing tasks, rather than academic-researcher writing tasks [Orr1999].

For example, the goal of reflective learning in a course can be achieved through the use of a Writing for Learning Task such as the weekly journal/diary. By locating this task in the taxonomy, the instructor can review prior work and construct a student writing assignment from the referenced model.

GENERAL ADVICE

In addition to writing tasks, many researchers offer general advice on the writing-for-computer-science problem. We have categorized this advice, along with citations in Table 4. Although the advice is wide ranging, some instructions are common:

- Give assignments a real world context
- Demonstrate that writing is important in industry
- Show parallels between the writing process and the software development process
- Require revision
- Conduct peer reviews of assignments
- Use minimal marking

With a decade working in the software industry and fifty years teaching writing, we have some additional advice on the writing-for-computer-science problem.

TABLE 1: WRITING FOR LEARNING	
Category	**Genre**
Critical	peer evaluation [Nelson2000] peer structured document walkthrough [DOE2002] portfolio [Ladd2003] partial revision of student paper [Haswell1983] critique of journal/magazine article [Bengston2005]. critique of programming language [Dugan2005a]
Persuasive	class debate [Egan1996] essay - discuss sides of issue and argue for one side [Egan1996] ethics paper [Kirzner2002]
Reflective	programming project reports [VanDegrift2004] response to reading assignment [Jacobson1989] summary of reading assignment [Jacobson1989] response to important course topic [Jacobson1989] response connecting two different lectures [Jacobson1989] weekly journal/diary [Fekete2000] weblog [Wikipedia2005] one-minute essay [Orr2005] replacement of traditional problem set with writing assignment [Anewalt2002]

TABLE 2: WRITING FOR ACADEMIC COMMUNICATION	
Category	**Genre**
Scholarly	research proposal [Zobel1996] research paper [Zobel1996] magazine article for internal publication [Côté1992]
Oral	class presentation [Polack-Wahl2000]
Technical	explanation of code/data structure/algorithm [Anewalt2002] laboratory reports[Parker1995]

Use clear, concise, and detailed models for writing tasks

If a picture is worth a thousand words, so is a model. A model is an example of the type of writing you want the students to produce. The students can note length, neatness, and visual format. They can see examples of organization, sentence structure, phraseology, and word choices as you point them out. For example: Persuasive memos have headings with To:, From:, Date:, and Subject: [Alred2003]. Letters have traditional parts in specified places [Alred2003]. Progress reports may be in memo or letter format with bold section headings and bulleted subheadings [Alred2003]. Formal reports have many parts in addition to the body. Models will show these parts and the parts within the body, including the placement and labeling of charts and graphs [Alred2003]. Summaries accompanied by a copy of the original article can provide insight into the selection of ideas to be included and techniques for including them logically and smoothly [Bengston2005]. Models from the workplace when available can be helpful in motivating students because these models have an air of authenticity.

TABLE 3: WRITING FOR INDUSTRIAL COMMUNICATION	
Category	**Genre**
Team	meeting [Alred2003] - agenda for meeting[Alred2003] - minutes for meeting [Alred2003] presentation [see Writing for Academic Communication/Oral] email [Alred2003] posting to newsgroups/bulletin boards/listserves [WJHCS2005] memo [Alred2003]
Customer	survey/questionnaire [Salant1994] white paper [Orr1999] presentation [see Writing for Academic Communication/Oral] website [Alred2003] user manual/help [see Writing for Industrial Communication/Project Management] proposal/requirements [see Writing for Industrial Communication/Project Management] bug report [see Writing for Industrial Communication/Project Management] executive summary [Writing for Industrial Communication/Project Management]
Project Management	weekly status report [Alred2003] bug report [Pfleeger2001] core documents: - proposal [Almstrum2005] - project plan [Almstrum2005] - requirements [Almstrum2005] - design [Almstrum2005] - test plan [Almstrum2005] - user manual/online help [Bremmer1999] executive summary [Alred2003] project post-mortem report [self-reported]
Career Management	resume [Alred2003] letter [Alred2003] - acceptance letter [Alred2003] - resignation letter [Alred2003] - interview followup letter [Alred2003] job description [Alred2003] employee appraisal [Orr1999] team mission statement [Orr1999]

Orient students to basic rhetorical needs

Rhetorical contexts should be set up for writing tasks based on models. Each task should designate a purpose, the role of the person who needs the information, and the persona of the students who are responsible for responding to the task. Students can assume the roles of the people who must produce written documents, namely programmers. For example, a model for a proposal or for a persuasive memo or letter can be used to show the need for a predetermined purpose and reader and offer words and phrases to use when trying to persuade a designated reader.

Teach types of definitions

Any one or a combination of definitions can be found in models: categorical, operative, contextual, stipulative, or metaphorical. Technical documents frequently need to define terms. Students can note the reason for choosing a type of definition and use that type when needed in a response to a writing assignment. For example, frequently they will need to categorize a term: "Testing is the filter . . . ." Sometimes they will need to tell how something operates or what it does: "Modularity provides . . . ." If the term has a meaning peculiar to the field or context, they need to specify: "In Computer Science, robustness is concerned with . . . ." If they use a term in a special way, they will need to stipulate its use: "In this paper, we will use the term *reliability* for . . . ." Occasionally they may need to use or interpret a metaphor: Frederick P. Brooks, Jr., refers to current-day large-system programming as a "tar pit" [Brooks1995].

Teach structure as needed

The first feature apparent in a well-prepared document is the neatness on the page. The second feature is the ease with which the reader can skim the headings and lists. Students will note this, but they may not know how to replicate it. Therefore, parallel structure must be taught: parallelism constructed with individual words, modified words, phrases, and clauses. The instructor may start by having students note the parallel items in the model and then refer them to explanations and illustrations of the various possibilities in any good handbook of the English language. For example: The categories in the "Writing For Learning" chart are all adjectives, adjectives that are understood to modify the word *writing*. The headings under "General Advice" are verbs in the imperative form: "Use clear, concise, and detailed models . . ."; "Orient students to . . ."; "Teach types of definitions . . ."; etc. Sentences introducing coordinate ideas begin with the same structures and sometimes the exact same words: "To address obsolecence . . ."; "To expand the extent of our literature . . ."; "To assist the overworked-small-college-computer-science instructor . . .". Series within sentences are in parallel forms: all nouns: "organizations including ACM, IEEE, ABET, CSAB, and NACE" and all adjectives: "clear, concise, and detailed." Any items connected by *and* have been put in parallel form: "We discuss the application of . . . and advice to . . . ."

Teach grammar as needed

The instructor may review the model in advance for grammatical structures that appear frequently and give a pre-lesson or let students write a draft, and then identify errors and give mini-lessons to help students avoid making these errors. For example, students should know the difference between active and passive voice and the reasons for using each. Instructors should know that when students use the passive voice, they are likely to start sentences with dangling modifiers. A handbook can be used to illustrate this grammatical error with sentences that the students will not forget, such as "By making this decision, our final product will not be as flexible." They will note quite quickly that the sentence has the final product making the decision instead of a person. The principle can be applied to sentences in student work where the error is less apparent.

Use the face-to-face conference

Face-to-face conferences can be used to assist students who have a writing assignment in progress. In these conferences, the instructor can point out inconsistencies between the model and work in progress and query the writer about various aspects of this work. For example: Who is your intended reader? Will that person know what this word/phrase means? What is the relationship between the idea in this sentence and the sentence that precedes it? Have you expressed this relationship? How can you express this relationship? Reword the sentence? Add a transition? Add a sentence?

Recommend the Writing Center

Most colleges and universities have Writing Center (WC) services available to students. The tutors in these centers are especially effective in helping ESL students with the English idiom, tenses, and word choices. However, the tutors can also be very helpful to native speakers of English.

Faculty who want to recommend the WC to their students should talk with the person in charge about the specific services. Some WCs are open to having faculty submit assignments for their students. Some WCs have forms for communicating with the instructor about a student who comes. However, whatever services the WC offers, students should take an assignment handout to the WC along with their responding drafts. Sending students to the WC raises the awareness of the instructor to the degree of clarity of the assignment. Anticipating a WC visit makes a student more conscious of articulating whatever problems he or she is having. Interacting with a WC tutor gives the student a clearer sense of what he or she is actually saying, needs to change, needs to add, or needs to delete.

APPLICATION OF TAXONOMY AND GENERAL ADVICE

We have applied the taxonomy and general advice of this paper to writing tasks in software engineering, programming languages, and database courses.

During the Spring 2005 semester, we (the computer science professor and the writing professor) co-taught a writing-intensive, senior software engineering course which had two goals: to prepare students to design a program for a real client with a real need and to prepare students to produce the documents required for this project, as well as several persuasive-type documents critical to team-based software development. To meet these goals, we assigned students by groups to real clients, found and adapted appropriate models [Almstrum2005] from our taxonomy, taught needed structure and grammar from a handbook *Technical Writer's Companion* [Alred2003] in classroom situations, and had the groups meet outside of class with the writing instructor (for peer and instructor evaluation) to read and revise each document until it reached an exemplary level.

To determine if student writing actually improved, we took a common approach to evaluation [Reed-Rhodes1998]: We surveyed software engineering course graduates for their attitudes toward writing. Using targeted questions with Likert scale responses (1=strongly agree, 2=agree, 3=somewhat agree, 4=disagree, 5=strongly disagree), we

surveyed graduates for their attitudes toward writing improvement (avg. 2.63, std. dev. 0.96), writing task guidance (avg. 1.88, std. dev. 0.81), and industrial application (avg. 2.29, std. dev. 1.00). Despite limited sampling (n=8), the survey has encouraged us to continue using our taxonomy and advice and to continue surveying software engineering course graduates.

TABLE 4: GENERAL ADVICE

Category	Advice
Anxiety and Resistance	Recognize that CS instructors are anxious about teaching writing [Kay1998] Attack student anxiety and resistance towards writing [Beer2002] - understand underlying causes [Beer2002] - demonstrate importance of writing in industry [Beer2002] - show parallels between writing process and software development [Beer2002] - show persuasive writing is a valuable industry skill [self-reported]
Incorporation of Writing into Curriculum	Incorporate writing assignments at all levels of curriculum [Cunningham1995] Provide writing intensive writing experiences within selected computing courses [Cunningham1995] Give assignments a real-world context[Cunningham1995] Incorporate active research into student work[Cunningham1995] Use short assignments to focus thoughts and as a basis for in-class assignments [Cunningham1995] Be abundantly clear about the expectations for the assigned paper [Anewalt2002] Vary the type of writing required if possible [Anewalt2002]
Collaboration	Encourage collaboration between computer science and writing instructors [Cunningham1995] Cooperate with technical writing students [Mengel2000]
Evaluation	Distinguish between syntax/mechanics problems and content/organization problems [Kay1998] Use hands-on, writing workshops [Pesante1991] Complete an evaluation sheet for each paper as you read it [Anewalt2002] Use cold-conferencing rather than formally graded drafts[Anewalt2002] Conduct peer review of writing assignments [Cunningham1995] Have students post something on an external bulletin board/newsgroup for comment [Cunningham1995] Use minimal marking[Kay1998] Use electronic portfolios[Ladd2003]
Basic Principles	Use early definition of terms [Brown1989] Use simple, parallel construction of ideas [Brown1989] Have consistency of format and word usage [Brown1989] Use conciseness and a minimum of redundancy [Brown1989]
Miscellaneous Concerns	Promote student familiarity with subject literature [Cunningham1995] Require revision Kay1998] Publish student work in a formal venue [Cunningham1995] Find out what others are doing [Pesante1991] Encourage broad and frequent reading in/out of the discipline[Kay1998]

With this experience using models, asking for early drafts, and recommending the Writing Center, I (the computer instructor) proceeded the during Fall 2005 to use a model adapted from several in our taxonomy [Johnson2006] when assigning a critique of a programming language in a programming languages course. The adapted model included a title page, table of contents, section and subsection headings with detailed descriptions, and a reference section with sample citations [Dugan2005a]. I encouraged the students to submit early drafts for instructor evaluation, and to use the college Writing Center for peer evaluation. The model, drafts, and Writing Center allowed the students to focus on learning and analysis rather than organization and content. The model also gave me criteria for grading the reports. Students told me that they actually enjoyed writing the reports, and I was very pleased with the results.

This experience convinced me that adapting models from the taxonomy, explaining how to use them, responding to early drafts, and recommending peer evaluation in the Writing Center produced much more desirable results than my previous method. During Fall 2004, I had taught the same course and asked for the same type of paper, giving students only a list of topics to cover [Johnson2006] and limited guidance. I had been very disappointed with the analysis, organization and content of the reports I received from the class, and I felt uncomfortable grading the reports harshly because I wasn't sure how the students were supposed to know what I had expected.

I had similar contrasting experiences with the writing assignments I gave in a database course during Spring 2003 and Spring 2005. A few weeks into the course I asked the students to write a memo to a fictional "boss" describing the advantages of a database system over a flat file system for a new medical application. The first time I taught the course I was appalled at the disorganized ramblings submitted by the students. As the red ink flowed from my pen, I realized that the students had no idea how to format and organize a business memo. How could they? The second time I taught the course, I gave the students the same writing task and supplemented it with a model [Alred2003], accepted early drafts, and encouraged use of the Writing Center. The resulting memos were very professional and demonstrated a clear understanding of database systems. The students knew what to write and I had criteria for grading.

CONCLUSION

In this paper we have addressed the computer scientist's lack of written communication skills crucial to success in the workplace. We addressed this problem with a goal-oriented taxonomy of clear, concise, detailed writing tasks and original general advice. We related our successful application of the taxonomy and advice to several computer science courses.

Experience applying the taxonomy and general advice to our courses has made us more comfortable assigning computer science writing tasks, but we realize that our approach has shortcomings. To address obsolescence we have created an on-line taxonomy that can be updated with new writing tasks and guidelines [Dugan2005b]. To expand our literature survey beyond IEEE and ACM publications, we intend to examine other work in the area of technical writing. To assist the overworked-small-college-computer-science instructor who may not have time to track

down the paper's many citations, we recommend the following core writing task guidelines: *The Handbook of Technical Writing* [Alred2000], *The Holt Handbook* [Kirzner2002], and *Writing for Computer Science* [Zobel1997].

In future work, we would like to explore techniques for grading writing tasks, identifying sources of/tracking student motivation, and facilitating computer technology in a college writing center. We would like to explore practical grading techniques that do not require a degree in writing or English. We would like to develop a survey to measure how a computer scientist's attitude towards writing changes throughout a career. We hope this survey will show students the importance of writing in a future career. Finally, we would like to know how the model for a college writing center changes with the integration of computer technology beyond the basic word processor. For example, what would happen if real-time collaborative software eliminated the need for face-to-face meetings at a writing center?

CITATIONS

Almstrum, V. (2005). *CS373: S2S Project Documentation Standards*. Retrieved Oct. 6, 2005 from http://www.cs.utexas.edu/users/almstrum/cs373/fa05/doc-stds

Alred, F., Brusaw, C., and Oliu, W. (2003). *Handbook of Technical Writing*, *Seventh Edition*. Boston: Bedford/St. Martin's.

Anewalt, K. (2003). A professional practice component in writing: a simple way to enhance an existing course. *Journal of Computing in Small Colleges*. 18, 3 (Feb. 2003), 155-165.

Anewalt, K. (2002). Experiences teaching writing in a computer science course for the first time. *Journal of Computing in Small Colleges*. 18, 2 (Dec. 2002), 346-355.

Beer, D.F. (2002). Reflections on why engineering students don't like to write -- and what we can do about it. In *Proceedings of the International Professional Communication Conference*. (Sep. 17-20, 2002), 364- 368.

Bengston, V.L. and MacDermid, S.M. (2005). *How to Review a Journal Article: Suggestions for First-Time Reviewers and Reminders for Seasoned Experts*. Retrieved Sep. 29, 2005 from http://www.ncfr.org/jmf/review_journal_howto.htm

Bremer, M. (1999). *The User Manual Manual: How to Research, Write, Test, Edit and Produce a Software Manual*. Grass Valley, CA: UnTechnical Press.

Brooks, F.J. (1995). *The Mythical Man-Month: Essays on Software Engineering, 20th Anniversary Edition*. Boston: Addison-Wesley Professional.

Brown, D.M. (1989). Writing good computer documentation. In *Proceedings of the International Professional Communication Conference*. (Oct. 18-20, 1989), 114-116.

Côté, V. and Custeau, G. (1992). An integrating pedagogical tool based on writing articles. In *Proceedings of the 23rd SIGCSE Technical Symposium on Computer Science Education*. (Mar. 05 - 06, 1992), 38-41.

Cunningham, S.J., (1995). Learning to write and writing to learn: integrating communication skills into the computing curriculum. In *Proceedings of the 7th Software Education Conference* (Nov. 22 - 25, 1994), 306-312.

Department of Energy (DOE). (2002). *DOE Systems Engineering Methods: Structured Walkthrough Process Guide*. Retrieved Sep. 27, 2005 from http://cio.doe.gov/ITReform/sqse/download/SW-V3-G1-0902.pdf

Dugan, R. (2005a). *CS323 Programming Languages: Language Report Model*. Retrieved Oct. 6, 2005 from http://www.stonehill.edu/compsci/CS323/ReportModel.doc

Dugan, R.and Polanski, V. (2005b). *A Taxonomy of Computer Science Writing Tasks*. Retrieved Oct. 6, 2005 from http://www.stonehill.edu/compsci/bdugan/taxonomy.htm

Egan, M. (1996). *Taking Sides: Using Taking Sides in the Classroom: Methods, Systems and Techniques for the Teaching of Controversial Issues*. Dushkin Publishing/Brown & Benchmark.

Fekete, A., Kay, J., Kingston, J., and Wimalaratne, K. (2000). Supporting reflection in introductory computer science. In *Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education* (Mar. 7-12, 2000), 144-148.

Haswell, R. H. (1983). Minimal marking. *College English*, 45, 6, 166-170.

Ladd, B. C. (2003). It's all writing: experience using rewriting to learn in introductory computer science. *Journal of Computing in Small Colleges*. 18, 5 (May. 2003), 57-64.

Jacobson, J.M. (1989). Response: an interactive study technique. *Reading Horizons*, (Winter 1989), 85-92.

Johnson, E. (2006). *CSCI 300 Programming Languages Fall 2002.* Retrieved Jan. 11 2006 from http://www.cs.xu.edu/csci300/02f/

Kay, D. G. (1998). Computer scientists can teach writing: an upper division course for computer science majors. In *Proceedings of the 29th SIGCSE Technical Symposium on Computer Science Education* (Feb. 26 – Mar. 01, 1998), 117-120.

Kirszner, L.G., and Mandell, S.R. (2002). *The Holt Handbook Sixth Edition*. Boston: Thomson-Heinle.

Kussmaul, C. (2005). Using agile development methods to improve student writing. *Journal of Computing in Small Colleges*. 20, 3 (Feb. 2005), 148-156.

Mengel, S.A.; Carter, L.; Falkenberg, J. (2000). A perspective on three cooperating courses. In *Proceedings of the 13th Conference on Software Engineering Education and Training*. (March 6-8, 2000), 265- 272.

Nelson, S. (2000). Teaching collaborative writing and peer review techniques to engineering and technology undergraduates. In *Proceedings of the 30th Annual Frontiers in Education Conference*. (Oct. 18-21, 2000), 1-5.

Orr, J. C. (2005). Instant assessment: using one-minute papers in lower-level classes. *Pedagogy*. 5,1 (Winter 2005), 108-111.

Orr, T. (1999). Genre in the field of computer science and computer engineering. *IEEE Transactions on Professional Communication*, 42, 1, (Mar. 1999), 32-37.

Parker, B. C. and McGregor, J. D. (1995). A goal-oriented approach to laboratory development and implementation. In *Proceedings of the 26th SIGCSE Technical Symposium on Computer Science Education* (Mar. 02-04, 1995), 92-96.

Pesante, L. H. (1991). Integrating writing into computer science courses. In *Proceedings of the 22nd SIGCSE Technical Symposium on Computer Science Education* (Mar. 7-8, 1991), 205-209.

Pfleeger, S.L. (2001). Testing the system: problem report forms. In *Software Engineering: Theory and Practice, Second Edition*. Upper Saddle River: Prentice-Hall.

Polack-Wahl, J.A. (2000). It is time to stand up and communicate [computer science courses]. In *Proceedings of the 30th Annual Frontiers in Education Conference*. (Oct. 18-20, 2000), 16-21.

Reed-Rhoads, T., Duerden, S.J., Garland, J. (1998). Views about writing survey -- a new writing attitudinal survey applied to engineering students. In *Proceedings of the 28th Annual Frontiers in Education Conference*, (Nov. 4-7 1998), 973-979.

Salant, P. and Dillan, D. (1994). *How to Conduct Your Own Survey*. New York: Wiley.

Thayer, R.H. and Dorfman, M. (1997). *Software Requirements Engineering*. IEEE Computer Society Press.

VanDeGrift, T. (2004). Coupling pair programming and writing: learning about students' perceptions and processes. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education* (Mar. 3-7, 2004), 2-6.

William James Hall Computing Services (WJHCS). (2005). *A Quick Guide to Newsgroup Etiquette*. Retrieved Oct. 3, 2005 from http://www.wjh.harvard.edu/wjh/newsgrp.shtml

Wikipedia (2005). *Types of Blogs*. Retrieved on Oct. 6, 2005 from http://en.wikipedia.org/wiki/Blog#Types_of_weblogs

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