Writing in an Upper-Level CS Course

Alan Garvey
Truman State University
100 E. Normal St.
Kirkville, MO 63501
660-665-9195
agarvey@truman.edu

ABSTRACT
This paper discusses the use of writing as a teaching approach for an upper level computer science course. In describing my experiences, I hope to encourage those schools/teachers who are considering incorporating writing into such a course. Many different kinds of writing are compared and contrasted. Attention is paid to practicalities of the writing process and of the demands made of faculty in these writing-based courses.

Categories and Subject Descriptors
K.3.2 [Computer and Education]: Computer and Information Science Education – computer science education

General Terms
Documentation, Management, Theory

Keywords
Writing to Learn, Writing Across the Curriculum, Writing in Computer Science Classes, communication skills

1. INTRODUCTION
I have been teaching Programming Languages as a writing-enhanced course at my public liberal arts and sciences university for 8 years. Over the years I have incorporated writing into the course in many different ways. This paper looks at some of those kinds of writing, plus a number of other ideas about how to incorporate writing into this or other courses.

To summarize, at my university a “writing-enhanced” course has desired outcomes relating to: cognition (using writing as a mode of learning, exhibiting critical thinking, showing audience awareness), process (engaging in deep revision, assessing their own writing, copy-editing as a regular habit) and product (being able to write clear, coherent, well-organized prose; demonstrating a command of syntax, style and tone; and exhibiting mastery of punctuation, usage and formatting conventions). One major goal in these courses is to facilitate critical thinking on the part of the student.

These desired outcomes leave quite a bit of freedom for the instructor to incorporate writing into the course in many ways.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGCSE ‘10, March 10–13, 2010, Milwaukee, Wisconsin, USA.
Copyright 2010 ACM 978-1-60558-885-8/10/03...$10.00.

Perhaps the most obvious is the traditional 20 page research paper and that is still a possibility, but there are many other ways to teach using writing.

The topic of writing in computer science classes comes up fairly often in the computer science education literature [2, 3, 5, 6, 7, 9] (and many other papers not cited here). Often there is an assumption that writing for computer science students should be taught in a writing class [2, 6, 7], perhaps one that is team taught with a writing teacher [2, 5, 7]. These papers are full of helpful ideas about how to teach writing, how to evaluate student writing and why writing is important for our students. The history of the writing across the curriculum (WAC) [1] movement as it relates to computer science is very interestingly discussed in a recent SIGCSE paper [5] and clearly my university’s writing enhanced requirement is a direct response to the WAC initiative.

In this paper, I attempt to illustrate how writing can be used for teaching in a standard, upper level computer science major course: Programming Languages. One motivation for writing this paper is as a response to that recent SIGCSE paper [5] which suggests that there is a distinction between the kind of writing suggested by WAC initiatives [1], which primarily are about using “writing to learn” (WTL), and the kind of “writing in the discipline” (WID) that is expected of computer professionals. These are seen as being so distinct as to require “bridge” assignments to get from one to the other. I don’t question the usefulness of the assignments described, but I question the discrete distinction made between the categories and would suggest that there is really more of a continuum. Perhaps my discomfort is just with the somewhat limited idea of what counts as “writing in the discipline”. Different kinds of writing are appropriate and useful in different contexts, but I think that practicing computer scientists in industry and academe are best served by being familiar with a broader selection of writing approaches and styles. And we need to recognize that even the most formal, technical writing often starts with an informal process of just getting the words written down.

2. KINDS OF STUDENT WRITING
Many kinds of writing assignments are available to instructors using writing as a way to teach a computer science course. This section attempts to define several categories of assignment and lists particular examples of each category in several subsections.

2.1 Summarizing Assignments
a. abstracts
b. reaction statements and questions
c. textbook summaries
d. lecture summaries
The idea with this category is to briefly summarize some kind of longer material. Being able to read a complex work, pick out the most important concepts and explain those in understandable language is difficult for many students. It often requires not just being able to understand and digest complex readings, but being able to fit them into a larger context of ideas.

Abstracts are formal, impersonal brief summaries usually of formal papers or texts. Learning to write in this formal, impersonal, unopinionated, factual style is useful in many other contexts in life. Being able to fairly summarize the position of another when describing your own position, for example, can be helpful.

Reaction statements are almost the opposite of abstracts. Here the focus is on the personal, perhaps informal reaction of the student to the material. If badly done this can just lead to rants, but carefully thought out reactions can facilitate deep thinking about a topic. I find that having the students come up with questions that they have after reading a paper can be a good way to get them to think critically and not take everything they read at face value. Questions can range from "what does this concept mean" to "why didn't they consider this alternate approach to this issue" to "how does this idea fit with that other idea we've discussed previously".

Textbook and lecture summaries are a way to force students to spend some time thinking about class-related materials. Among other things, these can be great aids in preparing for exams. For all but the lecture summaries, these kinds of writing can be required of the students BEFORE the material is discussed in class, as a way to get the students to come to class prepared to engage in meaningful discussions of the concepts.

2.2 Programs as Writing

a. documentation requirements in assignments
b. organization as a grading criteria
c. code walkthroughs
d. peer reviews of code

Presumably all CS professors should always evaluate all student programs as writing at some level. It is never acceptable for a student to submit undocumented, badly organized, incomprehensible code. But, in practice, we often focus on functionality over form.

The idea with focusing on programs as writing is to emphasize the importance of the human side of code readability. It is important that code compile and execute correctly (the computer side of readability) but it is equally important that code be understandable for other humans.

One way to approach this is through code walkthroughs. These are quite common in industry these days, but perhaps not as typical in the classroom. We often present students with what we consider to be high quality code and use that to explain and illustrate concepts. But with a code walkthrough the idea is to get the students to evaluate code critically and decide as a group whether a particular implementation is clear, well organized (and, of course, whether it is functionally correct).

A somewhat more unusual (in my experience at least) way to think about programs as writing is to adapt a standard Writing Across the Curriculum idea of peer review to programs. Have students produce "first drafts" of a solution to a programming assignment, then give each other feedback on those solutions. This approach has some interesting ramifications:

- It broadens students awareness by showing different ways of solving a problem.
- It helps those students who were seriously struggling with an assignment (and whose own peer evaluations will probably be quite negative) to see a way of solving the problem. This also brings to the fore an opportunity to discuss plagiarism and how it is okay to adapt the idea of another person, but not okay to "steal their code".
- It grounds more abstract discussions of desirable programming practice in particular examples: "here's why this particular organizational strategy doesn't work in this case", "this variable name is ambiguous in a way you might not have considered", "this particular code layout is irritating to read".

2.3 Reviews/critiques of the writing of others

a. review of CS books and articles
b. review/discussion of popular writing about technical subjects
c. reviews that compare/contrast multiple articles and introduce students’ own opinions (can be creative, but also requires some argument to defend the opinion)

Section 2.1 describes the summarizing of technical texts. This category is focused on more critical/review kinds of writing. In some ways this kind of writing is much harder. The writers have to both understand the other writing well enough to explain it, and then, in addition, add their own critical commentary on the writing.

One way to do this is to have the students read a technical article or book, say a CACM article or an article from a technical collection or a conference paper. Finding articles from these kind of publications that are at a level where an undergraduate can make sense out of them can sometimes be difficult, but such articles exist. Of course, finding articles that students can understand well enough to criticize effectively can be daunting.

Perhaps a somewhat easier way to approach this issue is to have students critique popular writing about technical issues. Articles from Time Magazine or Newsweek, or from the New York Times can be examined for their accuracy and clarity. Sometimes these articles are quite good, but often they are at least garbled if not just plain technically inaccurate. Getting students to realize that media coverage of something they are familiar with isn't always accurate can have the useful side effect of making them more critical of articles they encounter from outside of their expertise.

My favorite kind of writing assignment in this category has students compare and contrast multiple articles that address the same topic. For example, I give them two articles that address the issue of what programming languages will look like far in the future. Figure 1 shows an example of an assignment in this category. This particular assignment has worked well to focus discussion about the future of programming languages. Sometimes it is necessary to push students to do more than list issues from each paper and just say "I agree" or "I disagree". A
well written paper of this sort will demonstrate critical and creative thinking about a complex topic.

**Programming Language of the Future Assignment**

This writing assignment asks you to briefly (5-7 pages, 1500 to 2000 words) write about what you think the programming language of the future will look like, while also commenting on the thoughts of Paul Graham and Gregory Wilson on this topic.

One form of writing that is encountered quite regularly in magazines such as the New Yorker, the Atlantic, the New York Review of Books and others is the book review essay. In this form of writing the reviewer discusses a recent book (or, often, more than one book on a related topic) while also injecting their own, often strongly felt, opinions about the topic the books address. When this is done well the reader comes away with the feeling that they understand, at least vaguely, what the reviewed books are about, as well as seeing them from the unified perspective provided by the reviewer.

For this assignment, I want you to write in this format about the programming language of the future. The two works that I want you to discuss as part of your essay are The Hundred-Year Language by Paul Graham (also available in his book Hackers and Painters which is available through Safari) and Extensible Programming for the 21st Century by Gregory V. Wilson from ACM Queue Magazine. (Be sure to notice that the Wilson article has six sections, each with their own webpage.) I should come away from your paper with some idea of what Graham and Wilson think about this topic, as well as a clear idea of what you think.

You should write this paper for an audience of informed computer scientists (rather than a typical New Yorker reader, for example) who have not read the articles in question. You are expected to follow the conventions of academic prose. You are not required to cite other sources besides the two papers you are given, but, if you do refer to other sources, you should cite them using a standard academic citation style.

**Figure 1. Sample Assignment**

### 2.4 More creative kinds of writing

- a. speculative writing about the future
- b. taking a side and arguing for a particular perspective on a controversial issue
- c. any writing that involves opinions rather than just "facts"

The previous writing categories described here involve either paraphrasing or commenting on or otherwise responding to the words of other writers. Later categories often involve writing in a style that is carefully prescribed and not particularly open to creative expression. (Some may see this as a criticism of these writing styles, but that is not my intention here.) Both of these kinds of writing are useful, but in this category I’m gathering together the kinds of writing that require true creativity on the part of the student. In my experience, many students find this a particularly scary prospect.

The assignment shown in Figure 1 involves students reviewing papers that comment on what future programming languages might look like, as well as speculating themselves on the topic. That speculative aspect of the assignment probably best fits into the category I’m describing here. Well done responses to this prompt will not just comment on and slightly extend the ideas in the reviewed papers. Instead they will define whole new categories or perhaps find ways to organize the possibilities that open up new ways of thinking about them.

Another kind of paper in this category doesn’t directly bring in the works of others, but simply asks the student to take a side and argue for a position on a controversial issue. The best prompts in this area don’t have "right answers" and are open to disagreement. That might be because they involve ethical issues that society as a whole doesn’t agree on, or because they involve more aesthetic or subjective concerns. Still the best papers in this category won’t just be rants about the writer’s particular opinions on an issue. They will be organized, well argued stands that try to persuade others to agree with them on the issue at hand. An example of this is a paper that advocates the use of a particular programming language for CS 1, making persuasive arguments, but perhaps invoking the students’ own experiences as well.

I will limit myself to non-fiction prose writing for this paper, but obviously there is a whole category of creative technical writing that falls under the heading of sci-fi/fantasy. On rare occasion I’ve had students respond to a writing prompt by producing a dialogue or other short work of fiction. While this is often not an appropriate response to the given assignment, there are certainly real-world situations where such writing can be very effective. The trick is recognizing when more fictional approaches might succeed. One of the best replies to Searle’s Chinese Room argument that I have ever encountered is in an extra short story by Terry Bisson in which two sentient computers discuss these beings they encounter that appear to be "sentient meat". (quoted in [8], pg 96)

### 2.5 Scientific method research papers

Probably one of the most common kinds of writing assignment given in computer science classes is the classic scientific method paper, meant to be in the style of journal articles that introduce, say, a new algorithm or architecture, argue for why it is superior to previous state of the art, then conduct some kind of statistical experiments to quantify the scientific results. Since most journal and conference articles are (at least ideally) of this form, we see a need to include this kind of writing in our classes.

Having undergraduate students truly write a paper of this sort (including coming up with the innovation that the paper is about) is probably beyond what these students are able to do, especially in a classroom setting. One strategy to address that includes using this kind of writing for a simple lab exercise, where there is no need for true innovation, just investigation on the students’ part.
Another strategy I sometimes use is to have the students do a write up of this sort for an already well-known algorithm or data structure, for example, Quicksort or SkipLists. Even with this simplification, the experimental results section is usually still not achievable, due both to the time constraints and students’ statistical abilities. But we can talk at length about what such a section SHOULD look like and getting them thinking about these issues, preparing at least a few of them for the time in graduate school when they might be ready to make such a contribution.

2.6 Interdisciplinary writing
Along with the commitment to writing in my university’s curriculum there is a strong commitment to exposing students to interdisciplinary thinking and requiring them to engage in this kind of thinking themselves. Every student is required to take a junior-level interdisciplinary seminar that looks at a particular topic from the integrated perspectives of multiple disciplines. These classes have few prerequisites (other than junior standing) and are not discipline-specific, so they have to be understandable by a general student audience. For many years I have been teaching a kind of introductory cognitive science class that looks at human and (perhaps) machine intelligence from the perspectives of artificial intelligence, psychology, neuroscience, philosophy and other disciplines. This class is regularly taken by students who are CS, math, business, psychology and several other majors.

All of the kinds of writing described above are potentially useful in this kind of context. Since these classes often tend to involve readings of primary sources (rather than the use of traditional textbooks), the various summarizing kinds of writing, particularly abstracts, are especially useful. I regularly have students do 500 word abstracts of a reading, and let them hold on to these abstracts to help them organize their thoughts during class discussions of the paper.

Another kind of writing that tends to be particularly useful in these classes is explanatory writing that gathers information from multiple sources and synthesizes it into a coherent discussion of a topic, preferably from the perspectives of multiple disciplines. Doing this requires students to attain a certain level of expertise in disciplines with which they are only passingly familiar. (This is not the place to describe the demands on faculty in teaching such courses.) But, as I regularly tell my students in classes, "life is interdisciplinary". Even in their future careers that will presumably be at least somewhat based on what they studied in school, they will be required to analyze and synthesize material from outside of their immediate areas of expertise. And in regular living they will constantly be required to make decisions that weigh inputs from various perspectives and ways of thinking.

2.7 Team-based writing
As with interdisciplinary writing, the issues here are somewhat orthogonal to those of previous categories. Since most software development these days is done by teams, CS students can expect to often be part of a team that is asked to perform tasks that frequently involve writing.

In particular, software specification and design involves extensive documentation. Code itself continues to require documentation and organization. Code requires programmer-level documents written for those trying to extend or maintain the code. And, of course, applications require user-level documentation, perhaps written by a professional writer, but still requiring significant inputs from the software developer.

One issue that regularly comes up in team-based writing is trying to produce a document that fairly distributes the writing workload, while still not sounding like it was produced by 10 different people. One way to approach this issue is to have strict and detailed style guidelines that prescribe as much as possible how the writing should be done. This is often most easily accomplished for software documentation, but it is also useful for more general audience documents.

I think it is a useful exercise for every student to be required, at least once in their undergraduate experience, to write to a very detailed and prescribed standard. And then have their writing mercilessly reviewed with respect to that standard. This is not pleasant or enjoyable for anyone involved, but is a very real-world kind of assignment.

3. THE WRITING PROCESS
This paper is primarily focused on kinds of writing assignments that can be particularly useful in an upper level computer science major course. But the writing process students go through is at least as important as the writing products that they eventually produce.

Unfortunately, in my experience, for many students the perceived optimal writing process is to sit down the night before a paper is due and crank it out in one extended writing session. In fact, the ability to churn out 5 pages of reasonably decent prose in an evening probably correlates strongly with being a successful college student. And in one sense there is nothing wrong with this. Being able to produce grammatically correct, reasonably well organized prose quickly is a useful life skill.

The main problem with this process is that it isn’t conducive to the kind of deep, critical thinking we would like our students to engage in. Texts written quickly tend to reinforce the student’s ingrained beliefs and ideas, rather than getting them to think things through in a fresh way.

But there is at least a partial solution to this problem: in a word, revision. The idea is to have students produce multiple drafts of a text, revising earlier drafts to produce later drafts, based on feedback received from the professor or from peers in the class or from an institutional Writing Center or from wherever such feedback can usefully be provided. (Some of my students have been successful with having a parent provide feedback on drafts of papers. While I would never require this of all students, for some it can be quite helpful.)

In my experience, there are two major difficulties getting students to engage in deep revision. One issue is the student who produces a “rough draft” that is nowhere near complete, perhaps just a short outline of what the final product will be. If there is nothing to be reviewed, it is unlikely that these students will get useful feedback. The second issue is the student who produces a strong first draft and gets feedback that basically says “this is great”. To help avoid the first case, I usually tell students that their “first draft” (I studiously avoid the term “rough draft”) will not be graded, but that their completed paper will receive a lower grade (regardless of quality) if they do not hand in a serious effort for the first draft. The second issue is harder (for me at least) to address. If a student gives me a first draft that is essentially an “A” paper, what can I realistically ask them to do? Fortunately
(or perhaps unfortunately?) this problems doesn’t arise as often. Sometimes I will use this as an opportunity to suggest that the student might want to engage with the same topic in a later, longer paper assignment or perhaps consider the topic for their senior capstone project.

One way that the writing process can go drastically wrong is when students plagiarize. On occasion I have had students argue that, if someone else already said it better, why should they be required to rewrite an idea in their own words? But usually when students plagiarize it is for much more mundane reasons such as lack of time management skills, laziness or fear of a lower grade. The only things I will say about plagiarizing is that it has to be dealt with severely (meaning, for example, that the penalty needs to be worse than just a zero grade on the plagiarized assignment) and that the earlier it is caught the better. Catching plagiarism early is greatly facilitated by the revision process. If I see something in an early draft that looks suspicious, I immediately point it out to the student. More often than not, this makes the issue go away before any more significant response on my part is necessary.

4. FACULTY EVALUATION OF STUDENT WRITING

One question/concern that sometimes gets raised about having students do the kind of writing described in this paper is the expectations it makes of faculty. The two concerns I hear the most are faculty who feel unprepared to evaluate writing and faculty who see evaluating writing as requiring huge amounts of time and energy.

One response to the first issue is that some schools, at least sometimes, seem to encourage team teaching of these courses with both a computer science instructor and a writing instructor [2, 5, 7]. I think team teaching is great and I am all for it, but, at least at my school, the resources are rarely available to support it. And, again in my experience, it isn’t necessary. I’m teaching Programming Languages, not a writing class. I’m using writing as a way to get students to learn computer science.

One aspect of this concern is that most computer science professors are not trained copy editors. We don’t know the correct terminology for the kind of writing errors we encounter. But my response to this is that copy editing is NOT what using writing to teach is about in any significant way. On those (relatively rare) occasions when I encounter a grammatically incomprehensible paper, I will usually do simple copy editing of a page or two, to make sure the student understands what the issues are, then either have them copy edit their own work or send them to the Writing Center for detailed assistance.

This relates to the second concern mentioned, that of faculty time and energy. I find that interesting, creative papers that explore innovative solutions through writing are much more enjoyable to evaluate than twenty-five Java programs that are all attempting to do exactly the same thing. A good paper induces more energy in me, the evaluator, than the evaluation process consumes.

But, realistically, it does take time to carefully evaluate student writing. Where does that time come from? One answer is that I don’t carefully evaluate ALL the writing a student does for the class. For example, some abstracts might get careful evaluation and detailed feedback, while some get cursory checks. For example, some first drafts might be evaluated by peers (or the Writing Center) rather than by the professor. Also, writing assignments sometimes replace other assignments or exams in the class. Finally, my university has made a commitment to make Writing Enhanced classes have an (at least somewhat) reduced enrollment, so there may be fewer students to evaluate.

5. CONCLUSION

Writing is a life skill that all educated people should have, including computer scientists. By incorporating many kinds of writing into an upper level CS class, it is possible to both teach the students the important content of the course and to improve their writing and critical thinking abilities.

6. ACKNOWLEDGMENTS

Thanks to those who helped me overcome my fears about using writing to teach computer science.

7. REFERENCES