ASSESSING WRITING IN COMPUTER SCIENCE BACHELOR OF SCIENCE DEGREE PROGRAMS

Sherri Harms, John Hastings, Marilyn Jussel
Department of Computer Science and Information Systems
University of Nebraska at Kearney
Kearney, NE 68849
harmssk@unk.edu

Abstract

Program assessment has become increasingly important to maintaining and enhancing quality within computer science undergraduate degree programs. In addition, the assessment process is designed to measure the degree to which graduates meet industry requirements for entry-level positions.

A key learning objective for computer science students is communication in writing to both non-technical and technical audiences. Computer scientists must follow guidelines in the traditional sense of writing, as well as writing as it applies to the field of computer science, especially in the use of programming languages.

This paper describes a writing assessment process designed and implemented for an undergraduate program in computer science. Direct assessment measures at the course-level are used to assess student writing in both the software development processes as well as writing in the traditional sense. This paper includes an explanation of how the assessment measures are used to evaluate course and program-level outcomes.

1 Introduction

The importance of writing in Computer Science (CS) education has been justified from a variety of perspectives. Standards such as ACM/IEEE Computing Curricula 2008 recommend writing as an important communication skill. National professional accrediting bodies such as ABET have emphasized the importance of integrating the teaching of effective written communication into CS Bachelor of Science (BS) curricula. Surveys of IT professionals conclude that computer science curriculum should be revised to provide more emphasis on communication skills [Simmons and Simmons 2010].

Despite the importance of writing stressed by organizations like ACM and ABET, CS BS graduates lack written communication skills crucial to success in the workplace [Dugan and Polanski 2006]. Few CS BS curricula include a core course that incorporates the development of communication skills in the context of computer science and, where these courses do exist, the focus of these courses is on general writing skills rather than oral skills or professional communication skills [Falkner and Falkner 2012].

Cilliers [2012] noted that "despite documented success on the many experiences of incorporating writing into the computing discipline, the fact is that the challenge is as relevant today as it was in the 1980s. There is sufficient evidence to suggest that agreement is yet to be reached on exactly how to integrate successfully writing skills into traditional CS BS curricula, thereby familiarizing computing students with an essential activity of their chosen profession."

Despite the acknowledgment of the importance of writing, one major challenge is that there is not agreement among these individual educators, administrators, or programs as to what constitutes effective teaching of writing [Kortsarts et al. 2010]. Additionally, many CS BS programs, such as the one at the authors' institution, cannot afford to offer a separate course that focuses exclusively on communication skills.

A number of CS educators have responded to institutional mandates for writing intensive courses, to accreditation requirements or have justified writing on its own merits as a critically important professional skill [Anewalt 2003; Cilliers 2012; Dugan and Polansi 2006; Falkner and Falkner 2012; Garvey 2010; Hoffman et al. 2006; Kortsarts et al. 2010; Ladd 2003; and Michael 2000].

Michael [2000] discussed how to modify a programming course to incorporate assessment communication skills. He explains, "The usual expectations of students in a programming oriented course must be expanded both in breadth and depth - students must be required to write and speak more and better than previously. Students must be convinced of the ultimate value of this endeavor."

Dugan and Polanski [2006] stressed that CS BS programs should give students 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; and conduct peer reviews.

2 Software Development as Writing

This paper submits that the coding process and design tools should also be assessed as writing activities since they are definitely writing tools important to the field. During the software development process, students use design tools to, in essence, analyze, plan and outline their solution to the problem. This analysis is then translated into a programming language and tested using the computer. The program goes through many changes and iterations before the final product is completed. Following the implementation phase, software design involves a cycle of testing and modifying the code. Finally, software enters the maintenance phase where it is modified to meet changing needs and goals. Virtually all of the software that computer scientists create could be included as writing, if coding and design are viewed as writing processes.
Programming languages have been identified as "formal languages." They have symbols, a grammar, syntax, and structure just as natural languages do. In fact, Chomsky [1956] created the idea of formal languages specifically as a means of communicating with computers. Granted programming languages are a subset of natural languages, the user must learn how to use them in a less forgiving environment.

Both software development and formal writing involve a logical, carefully organized exposition of complex ideas [Kay 1998]. Writing, like computing, is based on a hierarchy of abstraction levels from the lowest, typography and document design, through spelling, punctuation, syntax, usage, style and tone, organization, content and facts, all the way up to the document's ultimate effectiveness (which depends, as does software's ease of use, not only on the author but on the audience) [Kay 1998].

Anewalt [2003] gives the following explanation. "When a written assignment is given; the writer should think about the goals of the final document and organize his or her thoughts about the assignment. After a clear plan is constructed, the process of selecting language to communicate the ideas begins and the document is created. There is a stage of reading and modifying the document to ensure that the final product has the desired qualities. Finally, many computer science-related documents, like user's manuals, enter a maintenance stage in which they are modified as the software that they refer to is modified. Even beginning computer science students are familiar with the software design process and can see the parallels. By presenting this type of analogy to students, the relationship between the writing process and the software design process is made clear, and computer science students will feel more familiar with the writing process."

In addition, computer solutions should contain a substantial amount of internal documentation. Each module is typically identified as to its purpose, variable identification, connections (pre and post conditions), and data pertinent to its use in the program. Computer solutions are usually accompanied by various design tools, such as pseudo code, Data Flow Diagrams, and Unified Modeling Language (UML) diagrams.

Hoffman [2006] stated, "Indeed, programming is considered Writing in the Discipline (WID). It is a commonly used form of communication, either to a processor or another computer scientist, in which we expect fluency from others in the field". 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 compiles and executes correctly (the computer side of readability) but it is equally important that code be understandable for other humans [Garvey 2010]. The similarity between the writing process and the software design process is something that few traditional English literature courses recognize, but can make computer science students feel more connected to the writing process [Anewalt 2003].

3 Assessment Process

The university-wide assessment guidelines at the authors' institution specify that all major programs annually assess discipline-specific learning objectives [UNK Assessment Guidelines 2013]. Departmental assessment plans are reviewed every three years. Curricular mapping is also used to make sure the required courses in all programs cover the program's learning objectives, from beginning, intermediate and advanced perspectives. The university guidelines specify that the assessment measures should be able to support inferences about student mastery on specific outcomes, with clear and reasonable targets specified for each learning outcome/assessment measurement pair. University guidelines also specify that the assessment should be embedded within major courses. University assessment guidelines state that students should be provided with clearly stated expectations on the assessment measures and that students should be given guidance in meeting expectations. The guidelines also require that action plans be developed to address any findings that do not meet target goals.

At the author's institution, the computer science program student learning objectives are to: analyze problems related to computer science and determine the requirements for a solution; design solutions to problems related to the field using appropriate design technologies; implement solutions to problems related to the field; communicate and interact orally on a professional level with clients and fellow professionals; and write effectively for both non-technical and technical audiences. University guidelines specify that the writing assessment process should include assessing student communication skills at both the lower and upper levels, while the other learning objectives can be assessed at just the upper-level.

Because the computer science student learning objectives include writing for both technical and non-technical audiences, and because of the university requirement to assess writing at both the lower and upper levels, writing is assessed in two different courses and in two different ways. In the Data Structures and Algorithms course, taken during the sophomore year, student writing in the software development process is assessed. In the Senior Capstone course, student writing in the traditional sense is assessed. Michael [2000] noted that there are advantages to developing and assessing communication skills in an existing course: "The impact on class time and traditional course content is negligible. The writing activity is more relevant since it is directly tied to the programming the students are doing."

3.1 Writing Assessment Measures

3.1.1 Writing Assessment in the Sophomore-Level Data Structures and Algorithms Course

The writing assessment measure in the sophomore-level Data Structures and Algorithms course is the final programming project submitted toward the end of the semester. The particular programming language and environment may vary from semester to semester. For the past two years, the student projects have been Android applications. Students worked on their projects as individuals during the final month of the semester. The project has two parts: a proposal and the app itself.
Although not technically part of the assessment process, the students write a brief proposal describing an Android application they will create, and pretend that the instructor (the intended audience) is a potential customer/client that they try to impress with their "awesome" idea. The students are instructed that their app should be of an appropriate scale, but also feasible given the time constraints.

This portion of the project ensures that the students carefully think about their project and come up with an idea of an appropriate scale (and the scale of their app is something which is assessed). The students have worked on Android programming throughout the semester and hopefully develop somewhat of a feeling for the sort of project scale which would be reasonable during the available time frame.

To give the students some early project management experience, the students include a prioritized list of features that the app will include, including the most critical features that the app absolutely must have in order to function. Each of these features has an estimated completion date.

The second part of the project is the app itself, which includes both the app and the Javadoc documentation of the classes and methods. The writing assessment is conducted on this portion of the project. The students wrap up their project by giving a presentation to the class.

The delivered app itself is assessed. A minimum of five student samples are collected, or the samples from the entire class, if there are fewer than five students. If there are enough students, a maximum of one-third of the student programs are randomly selected for assessment. The assessment is completed by the faculty member teaching the class at the end of each semester, using the rubric shown in Appendix A. The target goal for this measure is that the average assessment score of all program samples assessed will be at least 2.25 out of a 3.0 scale, on the assessment rubric for the programming projects.

3.1.2 Writing Assessment in the Senior Capstone Course

For the senior capstone course, students write a term paper related to technological ethics topics covered in class discussions and readings. These topics include such areas as

privacy, freedom of speech, intellectual property, and crime. The papers are not intended to be a simple summary of the issues covered in class. Students are particularly encouraged to pick compelling topics and are allowed to focus on specific events if they are of sufficient depth for a term paper. The students are given the freedom to write various styles of papers including literature reviews, position papers, or case studies. In order to get additional practice with the writing revision process, students submit a rough draft for comment prior to submitting their final work. Additionally, the instructor recommends that students have their papers reviewed by the writing center prior to the submission of the rough draft.

A minimum of five student writing samples are collected, or the papers from the entire class, if there are fewer than five students. If there are enough students, a maximum of one-third of the student papers are randomly selected from each course for assessment. The assessment is completed by three faculty members at the beginning of each fall semester, using samples from the previous academic year. These samples will be evaluated using the rubric shown in Appendix B. The target goal for this measure is that the average assessment score of all writing samples assessed will be at least 3.0 out of a 4.0 scale, on the assessment rubric for the capstone course project.

4 Results

For this past year, on a rubric scale of 0-3, the average writing assessment value for students assessed in the Data Structures and Algorithms course was 2.06. This was the first year of implementation using technical writing as the assessment measure for the Data Structures and Algorithms course. The average fell below the target goal (of 2.25). However, while four of the selected samples provided acceptable to strong results, the overall low score was the result of the inclusion of the work of an extremely unmotivated and weak student who was given a score of 0. Without that score, the average for the course was 2.575, which met expectations. Because the target goal was not made, an action plan was developed to address this issue. In this case, the action plan developed stated that in the future, only those students who pass the course with a grade of C or better will be assessed. The faculty expects that this method will generally eliminate the effect of outliers skewing the assessment.

The average for the upper-level senior capstone course was 3.09, which met the target goal (of 3.0) on a 4.0 scale for this past year. The senior capstone course has met the target goal since the 2008-2009 academic year.

Writing assessment was established at the author's institution in 2007-2008. The results of the first year's assessment process indicated that many students in the senior capstone course had difficulty meeting the support expectation of the assessment measure, and the assessment target goal was not met during the first year. The action plan developed had the instructors modify the assessment measure to provide more clearly stated expectations and require students to submit drafts papers, and after revision, the final papers. These minor adjustments improved the average assessment values, which have met the target goals since then in this course.

The data is collected from all assessment instruments yearly, and minor adjustments are made if necessary. As part of a continuous improvement cycle, the outcome assessment data, the assessment plan, the curriculum map, and the evolving national standard curricula guidelines are all used to revise the existing program every three years. Through these processes, the faculty members study the issue and formulate recommendations when the assessment measures indicate that an objective is not being adequately met.

5 Discussion

As the authors' institution is a liberal arts institution with a fixed number of credit hours allowed for a major program, the authors have struggled with ways to incorporate writing into the CS curriculum, especially in the lower-level computer science course. The authors felt that a traditional research paper did not fit with the focus of the data structures and algorithms course material and took away from inside/outside class time that lower-level students need for learning to be computer scientists. The authors believe the assessment process described in this paper is a good fit for their program. It took careful thought to come up with something that would fit into the curriculum, meet the university assessment guidelines and also assess written communication as needed in the computer science profession.

In addition to the writing assessment at the authors' institution, both written and oral communication is intentionally integrated throughout the computer science program. Students complete hands-on projects in a majority of their courses. Communication expectations are embedded within these project assignments and noted in the CS BS program's curricular map of major course requirements. Students are required to submit user requirements, algorithms, and documentation (such as Javadoc) and to give oral presentations to their classmates where they provide an explanation of their project (including the code/data structure/algorithms used), similar to a real-world situation. Although the presentations are not part of the writing assessment, they are a natural extension of the type of projects that the students are doing. Students are giving presentations more often and earlier in their program of study. This holistic approach models real-life expectations and makes the instruction in communication skills more in line with the CS profession, as advised by Dugan and Polanski [2006].

Additionally, students now realize that they will have communication requirements throughout the entire computer science curriculum. The authors believe that by initially focusing on the software development process as a writing process, students experience less frustration with communicating in writing, even when writing in a traditional sense.

6 Conclusion

Incorporating writing for both technical and non-technical audiences across a computer science curriculum is important to meeting the national standards and industry demands.

This paper discusses the writing assessment process used at one university. This process assesses software development as a writing process in a lower-level course, and follows this with writing assessment in the traditional sense in a senior-level course.

It is the authors' hope that this paper will lead to a deeper discussion of this topic with other computer science programs, to both share what they have learned as well as to learn how other programs assess writing. In having these discussions, perhaps agreement can be reached on what constitutes effective teaching of writing for Computer Science Bachelor of Science degree programs.

References

Association for Computing Machinery and Institute of Electrical and Electronics Engineers Computer Science Joint Task Force, *Computing Curricula 2008: An interim revision of CS 2001*, acm.org/education/curricula/ComputerScience2008.pdf

Karen Anewalt. 2003. A professional practice component in writing: a simple way to enhance an existing course. *J. Comput. Sci. Coll.* 18, 3 (February 2003), 155-165.

Charmain B. Cilliers. 2012. Student perception of academic writing skills activities in a traditional programming course. *Comput. Educ.* 58, 4 (May 2012), 1028-1041.

Chomsky, N. 1956. Three models for the description of language. *IRE Trans. Inform. Theory* 2, 113–124.

Robert F. Dugan, Jr and Virginia G. Polanski. 2006. Writing for computer science: a taxonomy of writing tasks and general advice. *J. Comput. Small Coll.* 21, 6 (June 2006), 191-203.

Katrina Falkner and Nickolas J.G. Falkner. 2012. Integrating communication skills into the computer science curriculum. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (SIGCSE '12). ACM, New York, NY, USA, 379-384.

Alan Garvey. 2010. Writing in an upper-level CS course. In *Proceedings of the 41st ACM technical symposium on Computer science education* (SIGCSE '10). ACM, New York, NY, USA, 209-213.

Mark E. Hoffman, Timothy Dansdill, and David S. Herscovici. 2006. Bridging writing to learn and writing in the discipline in computer science education. *SIGCSE Bull.* 38, 1 (March 2006), 117-121.

David G. Kay. 1998. Computer scientists can teach writing: an upper division course for computer science majors. *SIGCSE Bull.* 30, 1 (March 1998), 117-120.

Yana Kortsarts, Timothy Dansdill, Mark E. Hoffman, Adam Fischbach, and Janine Utell. 2010. Writing intensive and writing extensive: a continuum for advancing writing in computer science education: panel discussion. *J. Comput. Small Coll.* 25, 6 (June 2010), 205-209.

Brian C. Ladd. 2003. It's all writing: experience using rewriting to learn in introductory computer science. *J. Comput. Small Coll.* 18, 5 (May 2003), 57-64.

Mark Michael. 2000. Fostering and assessing communication skills in the computer science context. In *Proceedings of the thirty-first SIGCSE technical symposium on Computer science education* (SIGCSE '00), Susan Haller (Ed.). ACM, New York, NY, USA, 119-123.

Vibha Sazawal, Sarah Schwarm, Barbara Goldner, Ed Gellenbeck, and Carol Zander. 2003. Assessment of student learning in computer science education. *J. Comput. Small Coll.* 19, 2 (December 2003), 39-42.

Chris B. Simmons and Lakisha L. Simmons. 2010. Gaps in the computer science curriculum: an exploratory study of industry professionals. *J. Comput. Small Coll.* 25, 5 (May 2010), 60-65.

University of Nebraska at Kearney Assessment Guidelines, http://unk.edu/assessment, accessed March 18, 2013.

Appendix A. Sophomore-Level Data Structures and Algorithms Course Writing Assessment Rubric

Criteria	Three points	Two points	One point	Zero points
Level of project	⅄ appropriate scope/difficulty	⅄ intermediate level project	⅄ novice project	⅄ below beginning level
Syntax	⅄ free of syntax or runtime errors	⅄ free of syntax errors ⅄ has minor runtime error(s)	⅄ free of syntax errors ⅄ major runtime errors	⅄ critical syntax errors
Solution /Logic	⅄ solves target problem using best approach	⅄ solves problem but not in best manner	⅄ does not solve problem	⅄ illogical or missing
Supporting documentation	⅄ complete ⅄ sufficient detail ⅄ no or minor grammatical errors ⅄ no extraneous facts	⅄ requires minor additions ⅄ easily corrected grammatical errors ⅄ some extraneous facts ⅄ some details omitted	⅄ requires major additions ⅄ requires major revision ⅄ significant details omitted ⅄ many extraneous facts	⅄ missing
Appearance	⅄ intuitive ⅄ uncluttered ⅄ professional appearance	⅄ mildly counterintuitive ⅄ slightly cluttered ⅄ requires minor revisions	⅄ counterintuitive ⅄ cluttered ⅄ poor appearance	⅄ missing

Appendix B. Senior Capstone Course Writing Assessment Rubric

	1 Beginning	2 Developing	3 Acceptable	4 Advanced
Audience, Purpose, and Vocabulary	Inconsistent audience Uses inappropriate vocabulary for the audience Uses disinterested vocabulary Support is inappropriate for the audience Lacks a purpose beyond the aim Conclusion merely repeats what was already stated	Audience level varies in a manner that is distracting Uses somewhat inappropriate vocabulary for the audience Vocabulary somewhat varied and lively Support is somewhat inappropriate for the audience Attempts to create a purpose beyond the aim Conclusion makes little effort to address audience and purpose beyond restating what was said	Audience level slightly varies but is not distracting Varies slightly from appropriate vocabulary Vocabulary is mostly varied and lively Support, examples, and ideas are mostly appropriate for the audience Successfully attempts to create a purpose beyond the aim Conclusion makes an effort to address audience and purpose beyond restating what has been said	Has a discernable audience, presumably, but not limited to academic Addresses audience with appropriate vocabulary level Vocabulary is varied and lively Support, examples, ideas are appropriate for the audience and purpose Has a discernable purpose beyond the aim Conclusion addresses audience and purpose—does not just repeat what is already stated
Organization	Has no thesis Lacks organization Lacks introduction, body, or conclusion Paragraph(s) unorganized, incoherent, and illogical Paragraph(s) undeveloped Lacks connection between ideas and paragraphs	Thought present, contains a weak thesis Somewhat organized around the thesis Underdeveloped introduction, body, and conclusion Paragraphs unorganized, incoherent, or illogical Paragraphs somewhat underdeveloped Ideas and paragraphs are not fully connected	Contains an effective thesis Mostly organized around the thesis Contains introduction, body, and conclusion Paragraphs mostly organized, coherent, and logical Paragraphs mostly developed Attempt to create strong connections between ideas and paragraphs	Contains a strong, effective thesis Organized around the thesis Fully developed introduction, body, and conclusion Paragraphs are organized, coherent, and logical Paragraphs are fully developed Creates and maintains strong connections between ideas and paragraphs
Assignment, Structure, and Format	Grammar, mechanical, or spelling errors make ideas unintelligible Does not follow appropriate physical formatting Does not meet assignment length or topic requirements	Grammar, mechanical, or spelling errors cause confusion Consistently simple sentence structure Lacks citations Lacks Works Cited/References/ Bibliography Makes little attempt to use appropriate physical format Makes little attempt to meet the assignment requirements in length and topic	Very few grammar, mechanical, or spelling errors Sentence structure is varied Citations are present Works Cited/References/Bibliography present Attempt to use appropriate physical format Attempts to meet the assignment expectations in length and topic	No grammar, mechanical, or spelling errors Sentence structure is purposely skilled and varied Citation are present and in appropriate format Works Cited/References/Bibliography are present and in appropriate format Appropriate physical formatting Meets the assignment expectations in length and topic.
Support	Contains no support	Very low level of support Support is somewhat illogical or lacks variety Support is too long or too short to be effective Support is not integrated or explained Support is irrelevant to the thesis Support is ineffective	Slightly lacking in support Support is mostly logical and varied Support slightly too long or short Support is mostly integrated and explained Support is mostly relevant to the thesis Support is effective	Has sufficient amount of support Support is logical and varied Support is of appropriate length Support is smoothly integrated and fully explained Support is relevant to the thesis Support is very effective