Fostering and Assessing Communication Skills in the Computer Science Context

Mark Michael
King's College
133 N. River St.
Wilkes-Barre, PA 18711
mmichael@kings.edu

Abstract

In accord with a college-wide assessment program at the author's institution, a required major course approximately midway through a student's college career forms the matrix for an intensive project which both develops and evaluates the student's communication skills in discipline-specific ways. For Computer Science majors, the project is a component of a junior-level Advanced Object-Oriented Programming course. Though centered about a semester-long programming project, it involves expectations, guidance, and feedback beyond what is traditional. This assessment instrument has a minimal impact on class time and course content, substantial impact on faculty and student effort, and tremendous impact on learning.

Keywords

Assessment, communication skills, technical writing, software presentations, advanced courses.

1 Introduction

Long a concern in industry, the maturation of our Computer Science majors' communication skills is increasingly becoming a priority in CS education. Note, for example, Criterion IV-15 (IV-16, respectively) of the latest CSAB/CSAC criteria:

> The oral (written, respectively) communication skills of the student must be developed and applied *in the program* [emphasis added]. [1]

In [4], David Kay describes an upper-division CS course devoted entirely to fostering students' writing and speaking skills *in the CS context*. However, many CS programs cannot afford to offer a separate course of that type.

Certainly, any of the excellent methods used in the aforementioned course can be incorporated in a standard CS course. This paper shows one way to achieve many of the goals enumerated in [4], but in a junior-level programming course. It confirms what that paper's title proclaims — "Computer scientists *can* teach writing" — but adds, ". . . in an existing course."

Modifying a programming course to incorporate an activity of the type described herein requires a certain investment:

- 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 given guidance commensurate with the increased expectations.
- Students must be given feedback in regard to the broader range of skills being evaluated. Furthermore, the feedback should be more detailed and personal.
- Students must be convinced of the ultimate value of this endeavor.
- Faculty must become comfortable with and adept at analyzing student performance in areas they may never have been trained to analyze.
- Both students and faculty must devote (even) more time outside of class to the course.

However, 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 programming the students are doing.
- The writing requirements can actually be used to enhance the student's software development process.

2 Institutional Context

King's College, a church-related, liberal arts college with about 1700 full-time undergraduate students, has long had an active assessment program. The program is founded on

two key principles [2,3]:

1. Assessment should be embedded in courses.
2. Assessment should measure and foster academic growth throughout a student's four years in college.

The key component of the Assessment Program is the Four-Year Competency Growth Plans, blueprints designed by each department to direct all facets of its majors' development. These plans outline expectations of students, criteria for judging student competence, and assessment strategies with regard to several basic academic skills, e.g., effective writing. Expectations of students increase from the freshman year to the senior year. More importantly, the continued growth and assessment of competencies are the responsibility of a student's major department. In the freshman year, all departments have similar expectations of their majors. In subsequent years, competencies are nurtured and evaluated in discipline-specific ways. By their senior year, History majors and Computer Science majors have developed very different writing skills.

In addition to continual assessment charted by the Growth Plans, there are two comprehensive, one-time assessment events which take a wide-angle view of a student's academic progress from the perspective of the student's major: the Sophomore-Junior Diagnostic Project and the Senior Integrated Assessment.[1]

The former is the subject of this report. It is a graded component of a required major course taken in the sophomore or junior year. Coming roughly midway in a student's college career, the Sophomore-Junior Diagnostic Project is a "checkpoint" in that it identifies anyone whose command of fundamental academic skills indicates a need for remediation. However, the Project is not a "filter" but, rather, a "pump" which impels all students toward greater facility in gathering and communicating information. The diversity of Project formats is illustrated in [6].

3 History of the CS Diagnostic Project

For nine years, the Sophomore-Junior Diagnostic Project at King's was identical for both Math and CS majors. Since both took a required Discrete Math course in their sophomore year, the Project revolved around a substantial expository paper based on an assigned topic related to that course. The Project was a semester-long process, involving library research, writing, revision, an oral presentation, and several student-instructor conferences. [5]

Subsequently, the Math and CS Projects went their separate ways. The CS Project now resides in a required Advanced Object-Oriented Programming course (primarily involving C++) taken in the student's junior year. At this stage in their development, students are expected to write a substantial program requiring many weeks of work and considerable design skills. Therefore, a "one-dimensional" semester programming project is used as the nucleus of a more ambitious, "three-dimensional" project which puts to the test programming, writing, and speaking skills.

[1] For CS majors, the Senior Integrated Assessment is administered as a component of the senior-year Software Engineering course.

4 Assessment Principles

Assessment is far more than assigning a grade. Two students who receive the same *score* on an assignment may exhibit very different strengths and weaknesses. A traditional grade or score might not distinguish between them, and it will *help* neither of them. The purpose of assessment is to promote growth. A score/grade in itself does not constitute an aid in a student's development. What the student needs is detailed feedback.

While the Computer Science education community has an excellent tradition of providing such feedback on code students have written, it seems to have lagged behind other disciplines in demanding, fostering, and evaluating the continued development of basic academic skills (notably writing and speaking) in the context of the student's chosen discipline throughout the student's four-year undergraduate experience. The CS tradition of feedback needs to be expanded to encompass a broader range of skills — skills which will also be important in future professional success.

1. *Students should be provided with clearly stated expectations.* This means a list of specifications, an explanation of how grades will be computed, and any evaluation forms which will be used. These all help to focus students' attention on the desired skills, thereby encouraging their growth in the desired directions.
2. *Students should be given guidance in meeting expectations.* This includes lists of dos and don'ts, e.g., "How (not) to give a presentation using a computer." But specific concrete examples of good practice are perhaps more important. For example, in our case, students are sent a memo before being expected to write one. One area where exemplars are important is internal documentation, since most textbooks' programs contain minimal comments.
3. *Students should be given a second chance.* Resubmitting an assignment is common in Freshman English classes, but much less common elsewhere. Even the best students have plenty of room to improve and can benefit from a professional critique of their work. There are two challenges, however. The first is to convince students that the first attempt is to be their best effort, not a "rough draft." The second is to decide how to weight the two attempts. Some professors in other disciplines attach no point value to a first attempt. Our solution is to weight the final version twice as much as the first version.
4. *Students should be given detailed feedback in person.* At least an hour should be devoted to each student, even the best. Obviously this represents a substantial commitment of time on the part of the instructor, but it

is well worth it. For more than a decade, the conferences at which the students' first attempts are discussed have been the most intensive and productive learning experiences to which the author has been a party. "Assessment as learning" truly comes alive!

5. *Students should be referred for remediation when necessary.* Even though the Sophomore-Junior Diagnostic Project is a "pump" and not a "filter," an important objective is to make sure students have sufficiently developed the broad range of academic skills. Therefore, when a student communicates at an unacceptable level, the deficiency must be addressed. This process can begin as soon as the need is apparent, sometimes as early as the student's first memo. It ideally should involve those with training in teaching communication skills; at King's, students are sent to our Writing Center with specific directions on what skills need strengthening.[2]
6. *Faculty need leverage to ensure remediation takes place.* Despite the instructor's and student's best efforts, remediation must sometimes extend into the next semester. While this is the case for less than 10% of our majors, accrediting agencies increasingly demand evidence that such students are not merely identified, but also brought up to an acceptable level of proficiency in basic academic skills. Most students who find themselves in this situation freely agree that they are in need of continued help. However, the first such student in our department refused to believe his writing skills were inadequate. Consequently, for a decade, the College has allowed the grade of "Incomplete" (usually reserved for unavoidable catastrophes) to be awarded in the ambient course when the Sophomore-Junior Diagnostic Project indicates the student must be held to improving his/her writing skills. The carrot/stick of traditional course grades is the reason the King's Assessment Program is founded on *course-embedded* assessment.

5 Objectives of the Project

The dual goals of the Project are to develop and to judge the students' ability to communicate with a variety of people in a variety of ways. The following forms of communication take place in the Project:

form of communication	intended audience
internal documentation	co-workers
presentation/demonstration	
real-time, on-screen messages	the client
user's guide (readme file)	
project proposal (memo)	the boss
email	
formal reports	

[2] There is yet no analog for students with speaking deficiencies.

Certainly program comments and messages would be graded in any program, but now there is a greater emphasis on demanding proper grammar, spelling, etc. Though email communication is required during the semester, it is not graded with the same degree of rigor as the other components.

6 Format of the Project

The Project is far more than a programming project, a term paper, or both. It is a semester-long process that has many phases. A timetable is announced on the first day of class. Students are expected to abide by the timetable, with lateness at any stage incurring a penalty. What follows is a listing of the various steps in the trek.

6.1 Specifications

On the first day of class, students are given a description of the Project, including the motivation behind the Project and a timetable, as well as the evaluation forms that will be used. Since this is a multi-faceted activity, the specifications must carefully define the goal of each part, especially with regard to the level and tone appropriate for each of the different modes of communication.

6.2 Project Proposal

Very early in the course, the instructor sends the student a memo giving a more personal directive than the list of specifications. This provides students with a sample memo.

Students are given a few weeks to decide on a program and then write a memo to the instructor describing their projects. For the sake of efficiency, the graded, annotated memo constitutes the instructor's response. (A face-to-face meeting is scheduled if a proposed program is not suitable in scope for this level course.)

This memo also serves as an early-warning system for writing weaknesses. In fact, one student seemed at this stage to be destined for a grade of "Incomplete" but was able to concentrate on her weaknesses and cure them by the time the Project and the semester were over.

6.3 Progress Report

To discourage procrastination, the student is expected to send "The Boss" a weekly email update on what has been accomplished in the development of his/her program. A brief, formal progress report is submitted around mid-semester. At this point students must document that they are going through the software development process in the organized fashion about which we preach. Appropriate diagrams, such as CRC cards and UML, must be included. Students are told to assume their boss "once had their job" and, therefore, is well-informed on technical matters.

6.4 Preliminary Version of the Program, User's Guide, and Corporate Report

6.4.1 Corporate Report

A more comprehensive report covers all aspects of the program and its development. It defines the problem to be solved by the program and explains in detail how the program goes about solving the problem. Additionally, the student must describe the process by which the program evolved. A log of the student's weekly email reports is an appendix.

6.4.2 User's Guide

Although each program is supposed to be self-explanatory to the user, the student is required to write a user's guide. The purpose of this requirement is to give the student practice communicating with a technically unsophisticated audience. One of the most common difficulties students exhibit in the Project is, saying what they really mean. Thus, writing at a *lower* level may still represent a *higher* hurdle than writing for a colleague or supervisor.

The guide must state the purpose of the program, specify the system requirements for running the program, and explain how to install and use the program. In some cases, it may be appropriate to explain *how* the program works internally (at the algorithm level, not the implementation level).

The format of this Project is continually evolving. The user's guide was initially a printed document, is currently a readme file, and will probably take the form of on-line help screens in the future.

6.5 Conference on Preliminary Version of the Program, User's Guide, and Corporate Report

Even the best annotations on code and reports cannot say everything. Nor can they convey the instructor's passion about the importance of what is being assessed and his/her concern for the growth of the student. The Project is a very personal journey and deserves a one-on-one conference.

As pointed out in [4], some errors should be left for students to locate and fix on their own. This is particularly true if remediation is in order. (See §6.9 below.)

6.6 Final Version of the Program, User's Guide, and Corporate Report

The final versions of the program and written documents are submitted several weeks after the preliminary versions. In setting up a timetable, the instructor must allow time to evaluate the student's initial work and hold a conference with each student to discuss that work *before* the student can undertake revisions.

6.7 Presentation/Demonstration

The last computer lab of the course is devoted to student presentations. A generic expectation of all King's freshmen is the ability to use *PowerPoint*. A discipline-specific expectation of CS upperclassmen is the ability to demonstrate a piece of software. Both skills are put to the test as each student demonstrates *what* his/her program can do and explains *how* it does it.

While these presentations could be geared toward the client, we choose to have the programmer's colleagues form the target audience. This is so the explanation of the program's inner structure will be at a technical level consistent with the object-oriented design themes of the course.

Each student in the audience fills in the same evaluation form as the instructor. Even the presenter fills in one. Self-evaluations are often more critical and sometimes more insightful than the peer evaluations! Having the student reflect on his/her own performance has been recognized as a way to promote intellectual growth.

6.8 Final Conference

The conference regarding the student's performance on the entire Project is much shorter than the previous conference since most problems have been solved. The main piece of news is how the student's presentation was perceived. While the instructor's opinion is the primary determiner of the student's score on the presentation, the peer evaluations are combined, summarized, and conveyed to the speaker as a separate (sometimes-ambivalent) perspective.

6.9 Beyond the "Final" Conference

When the student's finished product indicates that writing deficiencies still persist, a grade of "Incomplete" is given until the student demonstrates adequate proficiency. Usually a further revision of the written materials suffices. Sometimes additional exercises have been assigned, e.g., a timed essay question on a subject related to a course the student was taking the following semester.

7 Administration of the Project

Since CS faculty typically have no experience "playing English/Speech professor," two crutches, viz., an evaluation form for the written reports and another for the presentation, were developed with the help of our English and Speech faculty.[3] Each form is broken into sections on content, organization, format, and mechanics and style. Each section has a list of questions, intended not as a true-false checklist but as a way to help the reader or listener focus on certain aspects and then write comments in spaces provided. The list can never be exhaustive; it invariably sparks observations made in the *spirit* of the questions listed. Some of the questions would apply to any report or to any oral presentation, for example (in the latter case), "Does the speaker articulate clearly?" Others are designed for this particular project, as in, "Do the *PowerPoint* slides clearly illustrate the structure of the program?"

[3] These forms can be obtained from the author.

Another important aid in administering the Project is a grading scheme (announced on the first day of class) that conveys to students that they are being graded on an entire process, not just a finished product. This is another motivational tool for those inclined to write code at the last minute or to treat the first drafts as *rough* drafts. The weighting of the various aspects of the Project is shown below. This represents one-fourth of the course grade.

Distribution of Credit for Sophomore-Junior Diagnostic Project

	Project Proposal	Progress Report	Preliminary Program and Reports	Final Program and Reports	Presentation and Demo of the Program
Content	4	6	2 (comments) 2 (messages) 3 (guide) 6 (report)	4 (comments) 4 (messages) 6 (guide) 12 (report)	5
Organization	3	4	3 (guide) 6 (report)	6 (guide) 12 (report)	5
Mechanics, Style, and Format	3	4	2 (comments) 2 (messages) 2 (guide) 4 (report)	4 (comments) 4 (messages) 4 (guide) 8 (report)	10
Design and Performance	—	6	18	36	—
TOTAL: 200	10	20	50	100	20

8 Findings

Students have found technical writing to be more challenging than writing they have done in their general education courses. Often, students who exhibit difficulties in writing during this project also exhibit difficulties in other courses.

However, the Project has occasionally revealed previously undetected problems, typically in organization, which did not show up in other courses when the student wrote mathematical proofs or lab reports, both of which tend to be of a fairly restricted structure. In explaining an entire topic, students have many more options and typically much more difficulty in organizing material, both globally and at the paragraph level. They also have trouble saying what they really mean when complicated concepts are involved.

Peer evaluations of the oral presentations have proved to be surprisingly valuable in several ways. First, by being involved in the evaluation process, students are more conscious of what is expected of them. Second, their perspectives provide a wealth of second opinions; often students catch flaws in presentation mechanics that the instructor might overlook while busy taking notes on content and technical accuracy.

The most surprising finding, however, is how widely sets of peer evaluations vary in their homogeneity; some presentations elicit similar responses from all in the audience, while in other cases it is hard to believe that everyone witnessed the same presentation. Peer evaluations, therefore, present several challenges to the instructor. What do you tell a student when there is no consensus among the audience? How does one judge the "beauty" of a presentation when different beholders see it differently?? What are the implications for one's own style of presentation???

9 Conclusions

The ultimate test of the efficacy of an assessment instrument is the positive impact it has on the further growth of students. The author has had the pleasure of viewing (admittedly anecdotal) evidence of that impact, e.g., by watching a previously inarticulate student being interviewed on TV or by reading an internship report from a vastly improved writer. A recent, unsolicited testimonial from an alumnus said: "P.S. Thanks for making us write all those memos to you in your classes. . . . I seem to write them just about every day."

When the King's College Assessment Program is itself evaluated periodically by faculty from all departments, the Sophomore-Junior Diagnostic Project is consistently the most highly regarded component of that Program. This is due to the pivotal role the Project plays in stimulating growth and, consequently, giving faculty and students alike a substantial return on their collective investment of extra time and energy.

References

[1] Computer Science Accreditation Commission of the Computing Sciences Accreditation Board. Criteria for Accrediting Programs in Computer Science in the United States (June 2000, Version 0.8, January 30, 1999) Online. Internet. [Sept. 3, 1999]. Available WWW: http://www.csab.org/criteria2k_v08.html.

[2] Farmer, D.W. *Enhancing Student Learning: Emphasizing Essential Competencies in Academic Programs*. King's College Press, Wilkes-Barre, 1988.

[3] _____ Course-Embedded Assessment: A Teaching Strategy to Improve Student Learning. *Assessment Update*, 5, (January-February, 1993), 8 & 10-11.

[4] Kay, D.G. Computer Scientists Can Teach Writing: An Upper Division Course for Computer Science Majors. In *The Proceedings of the Twenty-Ninth SIGSCE Technical Symposium on Computer Science Education* (Atlanta, Feb 25-March 1). ACM Press, New York, 1998, pp. 117-120.

[5] Michael, M. Assessing Essential Academic Skills from the Perspective of the Mathematics Major. In *Assessment Practices in Undergraduate Mathematics*, B. Gold, S. Keith, and W. Marion, Eds. MAA Notes #49, Mathematical Association of America, Washington, 1999, pp. 58-60.

[6] O'Brien, J.P., Bressler, S.L., Ennis, J.F., and Michael, M. The Sophomore-Junior Diagnostic Project. In *Assessment in Practice: Putting Principles to Work on College Campuses*, T.W. Banta, et al., Eds. Jossey-Bass, San Francisco, 1996, pp. 89-99.