Can We Teach Introductory Programming As A Liberal Education Course? Yes, We Can!

Elizabeth V. Howard (howardev@muohio.edu)
Department of Computer and Information Technology, Miami University
Middletown, OH 45042

Abstract

Twenty years ago as I was completing my undergraduate studies in computer engineering and beginning my engineering career, employers were interested only in my technical or "hard skills" and not in the "soft skills" of writing, listening, presenting, promoting diversity, and team-building. Today's successful IT companies now realize that not only are soft skills important, but they are fundamental contributors to continued fiscal growth for the company and personal success for their employees. Likewise, people in traditionally non-technical careers now need both soft skills and hard skills such as basic computer competency and problem solving skills. Where better to interweave hard and soft skills than in an introductory computer programming course? We can use the principles of liberal education to broaden the knowledge of both the technical and non-technical student. The question is: How do we integrate liberal education into an introductory programming course?

Keywords: introductory programming, liberal education, soft skills, Visual Basic, group work

1. BACKGROUND

When reminiscing about computers, most people concentrate on the technological advances: the move from mainframes to servers to desktop PCs; the advent of e-mail and the internet; the growth in use of Personal Digital Assistants (PDAs); the development of wireless networks; and the advances in multimedia. However, another major shift in computer science is the variety, in both depth and breadth, of computer use. In the past, the only people who worked with computers were scientists and engineers who had received specialized training in how to program and use them. Now, many jobs require the use of a computer and not just for e-mail or word processing but also for more analytical functions such as forecasting and reporting. "The explosive growth of desktop computing has taken information technology out of the glass house and into users' offices." (Vitello 1997)

Because of this phenomenal growth in computer use, the computing professional needs to master a completely new set of skills. Companies now seek employees "with a broader range of skills, including not only technical knowledge, but also communication and other interpersonal skills." (Occupational Outlook 2002). Not only must 21st century computing professionals possess strong technical ("hard") skills, they also need a strong set of accompanying "soft skills." Ann MacLeod (2000) defines the following list as the "soft skills domain":

- ability to communicate effectively
- creativity
- analytical thinking
- problem-solving skills
- diplomacy
- flexibility
- change-readiness
- leadership skills
- self-awareness
- team-building skills
- listening skills

Computing professionals can no longer segregate themselves in their own world of computers but must be able to communicate effectively with other technical staff and with non-technical end-users. "Like it or not, technical people must interact with non-technical users more frequently than ever before. And because business has become so critically dependent on technology, IS professionals must speak the same language as their colleagues in finance, sales, marketing, and human resources just to do their jobs." (Vitello 1997).

Similarly, professionals and support personnel in non-technical careers must have both computing skills and critical thinking skills. Many non-technical end-users of

software applications are now expected to create specialized reports and analyze information, and must have a stronger technical understanding of what is involved in software development. This confluence of hard and soft skills in the workplace demands that educators address the related issues and connect these skills. A first course in programming is a natural place to start, as will be demonstrated in the following discussion.

2. WHY LIBERAL EDUCATION IN AN INTRODUCTORY PROGRAMMING COURSE?

A purely programming course may seem daunting to non-technical students. They may not be comfortable with many of the analytical and problem-solving activities that are common experiences in technical courses. Incorporating traditional liberal education requirements into a programming course can help non-technical students see how the topics and skills from this course connect to other courses in their program of study. These students may have already been exposed to group dynamics, collaborative work, writing, oral presentations, and interpersonal communication. However, they will have new experiences where they will actually develop software and gain a clearer understanding that software is more than just an icon on their computer's desktop. A programming course shows them the effort and critical thinking skills necessary to develop and use software applications. Incorporating traditional liberal education requirements into such a programming course can help non-technical students see how the topics and skills from this course connect to other courses in their program of study.

At my university, courses approved as liberal education courses must address four basic principles. Students should learn: the ability to think critically, to improve their ability to engage with other learners, to understand the context of a problem and corresponding solution, and to reflect and then act on what they have learned. These goals are consistent with the Association of American Colleges and Universities' Statement (excerpt) on Liberal Learning (1997):

> "A truly liberal education is one that prepares us to live responsible, productive, and creative lives in a dramatically changing world. It is an education that fosters a well-grounded intellectual resilience, a disposition toward lifelong learning, and an acceptance of responsibility for the ethical consequences of our ideas and actions. Liberal education requires that we understand the foundations of knowledge and inquiry about nature, culture and society; that we master core skills of perception, analysis, and expression; that we cultivate a respect for truth; that we recognize the importance of historical and cultural context; and that we explore connections among formal learning, citizenship, and service to our communities.
>
> The ability to think, to learn, and to express oneself both rigorously and creatively, the capacity to understand ideas and issues in context, the commitment to live in society, and the yearning for truth are fundamental features of our humanity. In centering education upon these qualities, liberal learning is society's best investment in our shared future."

By focusing on the goals of liberal education in a programming course, we can introduce technical students to the set of soft skills now needed by IT professionals. Students taking the course as a liberal education requirement (and not as the first course in an IT career) learn the creative and analytical thinking necessary to create software applications and to use a computer as a problem-solving tool. All learn the interdependence of technical and non-technical skills and how they enhance each other. Interweaving liberal education principles in an introductory programming course is beneficial to all students. The question that remains is: how do we do this?

3. HOW DO YOU TEACH AN INTRODUCTORY PROGRAMMING COURSE AS A LIBERAL EDUCATION COURSE?

Course Description

The technical material covered in this course is typical of an introductory programming course. Students analyze problems, formulate alternative solutions, implement solutions using Visual Basic, and assess the solutions' effectiveness. Students will design and implement between eight and ten programming projects outside the classroom, including a final project that can range from extensions of previously completed projects to elaborate three-dimensional fantasy games. Students also create small programming projects that they complete during a class laboratory session. In addition to the programming topics, students are also introduced to a survey of hardware and software concepts and terminology. As a liberal education course, it should also apply the previously described liberal education principles in course activities, assignments, and general approach.

My initial efforts at teaching this course were often disjointed and frustrating for the students and for me. I segregated the hard skills of problem solving and coding from the soft skills of communication, ethics, and group work. I grumbled about "those liberal education requirements" imposing on my time to teach my students the "real" skills and concepts that they needed. Because of my dissatisfaction with the disconnected presentation of the liberal education requirements and technical content, I was forced to reflect upon my own experience as a new engineer and I realized that I had

been more surprised by the demands of my job for soft skills than I had ever been by the technical knowledge that was expected. My undergraduate education had prepared me well to tackle new technical problems but had not prepared me for the amount of writing, presenting, listening, and contemplating ethical dilemmas that I would face. With this realization, I began to try to incorporate the liberal education principles that mirrored these skills into course activities so that students could find a connection between those principles, computer science, their careers, and hopefully, their lives.

Activities and Assignments

The following sections describe some of the activities and assignments in this course that I use to incorporate liberal education principles. In these descriptions, I include my intended goal(s) and some typical outcomes. For each category, I include the liberal education principles (as defined by my university) that are addressed.

Personality Profiles (Reflecting/Acting and Engaging With Other Learners)

At the beginning of the semester, I briefly introduce the concept of personality types based on the work by Myers-Briggs (Myers & McCaulley 1989; Keirsey & Bates 1984). All students then complete an online questionnaire (Humanmetrics) to determine their Myers-Briggs personality type. In the associated writing assignment, I ask students to comment on the accuracy of their personality description, describe potential benefits or challenges of working with my personality type, and tell me whether they think that knowing another person's personality type would be beneficial. Additionally, I ask them to describe their ideal university. This allows students the opportunity to express their opinion on their educational process and to reflect on how personality type affects peoples' choices and preferences. My goals for this assignment include providing my students with: the opportunity to improve their self-awareness, practice communicating through both writing and discussion, exposure to ideas about how communication may be perceived by other people, and reflection on the impact of personality styles in the software applications that they design and use.

Throughout the semester, we revisit the importance and impact personality types have on group dynamics and how personality types can affect the development of presentations, documents, and software applications. An additional benefit to this exercise is that I also use the information from my students' personality types to broaden my teaching techniques and customize my interaction with students. For instance, if a student is an extreme introvert, then I may use the more informal in-class exercise time to interact with that student one-on-one to respond to questions or provide further instruction. The papers that I receive for this assignment are generally well-written and interesting with significantly fewer grammatical errors than in traditional research papers. When I asked my students why, they identified three main reasons: they were familiar and comfortable with the topic; they were interested in the topic; and they enjoyed the assignment. In fact, I often receive notes thanking me for assigning the personality profile paper.

Group/Teamwork (Critical Thinking, Engaging With Other Learners, and Understanding Contexts)

This class is designed so that students spend about one-third to one-half of class time working with other students in groups of various sizes. When I give a group assignment that I have intentionally designed so that students will need to ask me (or each other) questions to complete it, I assign specific groups. For simpler activities, I sometimes allow them to choose their group members with restrictions on size or I allow them to form groups without any restrictions so that they can work as an entire class, if they choose. My goals in providing a number of group and team experiences are to expose students to group dynamics from several different perspectives and to allow them to collaboratively create solutions to programming problems. I have found that my students greatly benefit from the collaborative exercises for several reasons: students often find it less intimidating to ask a classmate a question than to ask me, they realize that they often are not alone in their confusion, they learn that there can be multiple solutions to the same problem, and they learn to more critically evaluate solutions. As I walk around the class, I can answer questions in a less formal setting than in a lecture and provide targeted information and assistance. The experience of working in groups gives my technical students a preview of the work environment they will face as information technology professionals and demonstrates to non-technical students that the skills of cooperation and collaboration are used and valued in virtually all professions.

Realistic Design (Reflecting/Acting, Critical Thinking, and Understanding Contexts)

I try to create assignments that are as realistic as possible to allow students to see the process of software development. By creating assignments similar to actual software applications, students can better appreciate an effective visual interface and a program that produces the correct result. They also experience and understand how easily a program can produce the wrong answer without the programmer being aware that the answer is wrong (and also when the programmer *knows* that the answer is wrong). Since this course is taught using Visual Basic, students can also be exposed to the design of user interfaces. The user interface is the first thing that software users see, and it is easy to demonstrate to students that a good user interface can be as important as the operational features of the program. By performing software development, students also understand why software companies sometimes release software with errors, known and unknown. Not only do students begin

to understand the time and effort required to produce software applications, they also begin to view programs differently. Recently, one of my students (majoring in a non-technical area) came to class one evening and told us about seeing a software application at the tanning salon. Not only had she recognized the basic features of a Windows-based software application, she also exclaimed that she could have written that program. She had done what we all hope that our students will do – she had made the connection between her class work and her life outside the university.

Discipline-Specific Writing Activity (Critical Thinking, Understanding Contexts, and Reflecting/Acting)

For this course, the majority of writing activities are discipline-specific or "situated-learning exercises" (Gruber et al. 1999; Artemeva, Logie, & St.-Martin 1999; Dias, Freedman, Medway, & Pare 1999) in the form of technical writing assignments more representative of what is required in the information technology workplace. Technical writing typically focuses on a very specific purpose for an audience that has less knowledge than the author (Samson 1995; Pfeiffer 2001). To address this, students are asked to write step-by-step directions both for using software that they have created and also for using some feature of Visual Basic that we have not covered in the class. In the first assignment, the students learn the difficulty involved in explaining how to use software, even though they may have created the software themselves. In fact, they see that their familiarity with the product often hinders their ability to explain its function. In the second assignment, students must first teach themselves how to use the feature before they can create directions for someone else. This learning reinforces the need for writing clear instructions as they consult manuals, textbooks, and other sources to understand and learn a specific process. With these assignments, students gain experience in two critical areas: learning a new task independently and communicating that knowledge to others. This writing assignment covers many of the necessary skills noted by Ann MacLeod (2000), including: the ability to communicate effectively, to think analytically, to solve problems, and to be prepared for change. These exercises also provide an opportunity for students to incorporate and apply ideas learned in the personality profile exercise to help them create instructions that are meaningful to a broader range of people, and not just to people who think like the author of the directions.

Final Projects (Understanding Contexts, Critical Thinking, Engaging With Other Learners, and Reflecting/Acting)

The assignment of a "final" project allows for many of the technical topics in the course to be incorporated into one program along with the application of several liberal education principles. At the end of the semester, students design and implement final group projects. I impose a few technical requirements on the project, but the students are free to choose their application. My main objectives for the final project are to provide an opportunity for creativity, to allow advanced students the chance to explore more complicated projects, and to encourage students to work together outside of class. Some students create projects that are applications similar to what we have done in class; however, many students display incredible imagination, resourcefulness, and a willingness to work harder than requested. These projects have ranged from a technically complex program of a three-dimensional fantasy game to a creative and touching tribute to the victims of the September 11, 2001 tragedy. On the last day of class, students demonstrate their projects and the demonstration is a more realistic situated-learning environment because the student has transitioned into the expert and the audience, including the professor, knows less about their project than does the student. This project provides a fitting finale to the course, as it often implements many technical topics, and yet illustrates the connections and application of so many of the "soft" skills" provided by the liberal education aspects of the course.

4. CONCLUSION

The experience of transitioning a course with separate technical and liberal arts components into one where these components are tightly integrated has been very positive for my students and rewarding for me. Those students taking the course as the start of an IT career are able to learn how valuable writing, listening, presenting, working in groups, and other applications of liberal education principles can be in a technical field and begin to understand that these skills will be expected of them. On the other hand, students taking this course because it meets a the liberal education requirement develop an understanding of the analytical and critical thinking skills that are required to design and develop software, and learn how computer software can be a valuable tool that can be applied to a variety of fields. By integrating liberal education principles with technical concepts, students have an opportunity for increased self-awareness, creativity, collaboration, reflection, and critical thinking – just what the 21st century professional needs to succeed.

5. REFERENCES

Artemeva, N., S. Logie, and J. St.-Martin, 1999, "From Page to Stage: How Theories of Genre and Situated Learning Help Introduce Engineering Students to Discipline-Specific Communication" Technical Communication Quarterly, 8 (3), 301-316.

Association of American Colleges and Universities' Statement on Liberal Learning, http://www.aacu-edu.org/about/mission.cfm#liberal.

Beaufort, A., 1999, Writing in the Real World: Making the Transition From School to Work. New York: Teachers College Press.

Gruber, S., D. Larson, D. Scott, & M. Neville, 1999, "Writing4practice in Engineering Courses: Implementation and Assessment Approaches." Technical Communication Quarterly, 8 (4), 419-440.

Dias, P., A. Freedman, P. Medway, & A. Pare, 1999, Worlds Apart: Acting and Writing in Academic and Workplace Contexts. Mahwah, NJ: Erlbaum.

Humanmetrics. www.humanmetrics.com.

Keirsey, D. & M. Bates, 1984, Please Understand Me: Character & Temperament Types (5th ed). Del Mar, CA: Prometheus Nemesis.

MacLeod, Ann, April 2000, "The Importance of Soft Skills in the Current Canadian Labour Market." Prepared for: Sectoral and Occupational Studies division of Human Resources Development Canada. http://www.hrdc.gc.ca/sector/english /publications/skills.shtml.

Myers, I. B. & M. H. McCaulley, 1989, Manual: A Guide to Development and Use of the Myers-Briggs Type Indicator. Palo Alto, CA: Consulting Psychologists Press.

Occupational Outlook, 2002, Occupational Outlook Handbook 2002-2003 Edition, U.S. Department of Labor Bureau Of Labor Statistics. http://stats.bls.gov/oco/home.htm.

Pfeiffer, W. S., 2001, Pocket Guide to Technical Writing (2nd ed.). Upper Saddle River, NJ: Prentice Hall.

Reynolds, J. F., 1995, "What Adult Work-World Writers Have Taught Me About Adult Work-World Writing." Professional Writing in Context: Lessons From Teaching and Consulting in Worlds of Work. Hillsdale, NJ: Erlbaum.

Sadler, L. V., 1995, "Preparing for the White Rabbit and Taking It on the Neck: Tales of the Workplace and Writingplace." Professional Writing in Context: Lessons From Teaching and Consulting in Worlds of Work. Hillsdale, NJ: Erlbaum.

Samson, D. C., Jr., 1995, "Writing in High Tech Firms." Professional Writing in Context: Lessons From Teaching and Consulting in Worlds of Work. Hillsdale, NJ: Erlbaum.

Vitello, Jill, October 13, 1997, "Hard Facts on Soft Skills," Computerworld. http://www.computerworld.com/news/1997/story /0,11280,9869,00.html