DOI:10.1145/1498765.1498779

Michael Buckley

Viewpoint
Computing as Social Science

Changing the way computer science is taught in college by encouraging students to develop solutions to socially relevant problems.

I HAVE BEEN concerned with the decline in computer science enrollments nationwide,[a] and it began to increasingly bother me as I prepared for the fall semester last year. With each new freshman class I worry that fewer students are choosing a career in computer science, and it's no stretch of the imagination to think it's an image problem. Not only are we not getting the word out to high schools, but I believe we're losing students in the first year of college as well. And it's because we don't offer students the idea that computer science is social, relevant, important, and caring, and thus we lose their interest. There might in fact be studies that show this, or perhaps not, but it's something I feel.

I began my last summer break as I do almost every year, looking through textbooks for something usable for a fall course in programming. I teach Java, which is more than adequate for new programmers to learn everything good and bad about addressing a computer. What they learn is that computers never do what we want, but only what we tell them to do. And what we tell them to do in a freshman programming course is too often dull. Write to the operator, print out the results of a calculation, order some list, and all too often the message, calculation, and list are irrelevant. How can I get them interested? I'm determined at the start of every semester to have a batch of eye-opening, to-the-point, significant examples, lessons, and problems.

To learn more about David's story, see http://www.sociallyrelevantcomputing.org/.

PHOTOGRAPH COURTESY OF JUDY SCHINDLER

And so I look through the textbooks and the examples and sample programs, and I become aware of an obsession with animals. In the first 10 textbooks I've skimmed, I've learned how to count ducks, categorize puppies, separate cows from horses, manage a pet store, create a cyber-pet, add fish to a bowl, and so it goes. This can't possibly be the least bit interesting to a freshman who wants to learn computing.

I go back through the textbooks intentionally avoiding animal references and instead look for something else. I find games, plenty of them: Tetris, Othello, checkers, tic-tac-toe, even a good approximation of chess moves, which is wonderful if you come to college to play games. I'm not even sure what programming principle they're trying to teach, and in my best attempt at empathy with an incoming freshman, my eyes glaze over. I'm bored, and I have a vested interest. How can a student possibly find interest and relevance in this stuff? The texts rely solely on the student to be interested enough in programming to overcome the banality. We all know that practice is more fun than theory, but our attempts at practice aren't real.

I move on...let's see...a doughnut

a See http://www.cra.org/wp/index.php?p=105.

counter to show iterations, a pizza maker to construct a list, a lemonade stand to demonstrate databases. If I was a student, beginning these important four years, and I was taught programming via doughnut machines, I would quit and go do something important. Major in some field that had an impact. Even the sterile environment of pure mathematics has me counting and measuring planets and populations. Sociology and psychology would have me charting behaviors. Chemistry and physics have me connected to the environment. Computing as portrayed in the literature has me running a pet store, playing games, or eating. That would seem to be it. There isn't a textbook out of the 60 I have on my shelf that makes me see computing as socially relevant.

And so the message is just not getting out there. Students' firsthand experience with computers—their music and their phones—is accepted and reinforced by the image we portray in school—one of unrelenting banality and geekdom—and potential computer science students do not see themselves as having a greater impact.

At the University of Buffalo we have two senior-level courses that require teams to create real systems for real clients. They are introduced to the wider community of people with disabilities and told to make a difference. That's it. Those are the instructions. Improve the quality of life of someone less able than you. If you can't figure it out, you fail. So don't fail.

A group of students tacked a sign up at a school for handicapped children that said "Student Inventors Available Free. Is there something you need? Call us." And they heard from a mother whose daughter could not use a computer because she had no fine motor skills. She could move her arms but not her fingers. So they made her a trackball out of a basketball, and wrote games that use the wide swing of her arms as she rotated the basketball. It's not perfect, but they were immersed and involved, and they visited with the family and they delivered a prototype. And no one will ever convince them that computer science is not social science, because outside the world of trivia we feed students in their freshman year, it certainly is.

Now people in the community call us. That's how my students met David, who was 43, suffered a stroke at age 27, and hadn't been able to speak since. He communicated with his nurses by pointing to a sheet of paper that was taped to his wheelchair. It had letters, words, and short phrases, and after much practice, a nurse or therapist could almost decipher what he wanted to say. So our students transferred that sheet of paper to a tablet PC, and when David touches a word, the computer speaks it. How difficult is that? Easier than counting doughnuts. The night they delivered that system, David called me at home with his new voice and thanked me. And said he waited 15 years to speak on the phone. And the pictures I saw later of the event clearly showed students crying.

There isn't a textbook out of the 60 I have on my shelf that makes me see computing as socially relevant.

Every once in a while, a student will say "I can't find a project," and I tell them to read the newspaper or consult other news sources. That itself sounds banal but it's not: right below the surface of a news item, there is most likely a problem to be solved. Find someone or something in trouble, and save it. That's how we found the number-one killer of firefighters on the job: it's not fire, smoke, or Dalmatian attack; it's heart attack. And so now we have a system that monitors vital signs and displays the statistics on a 3D model of a fire scene as the firefighters traverse it.[b]

We have remote-controlled wheelchairs, videoconferencing for homebound and hospital-bound children, a light-and-sound system (the students call DISCO) that teaches cause-and-effect to autistic children, and many more systems constantly evolving. All of this technology and creative energy is at our fingertips, but to sample our craft in the popular literature, you would think we were cyber pets on one end, artificial intelligence on the other, and nothing useful in between.

So back to the textbooks and the freshman year. In the senior-level courses, you can see the difference between simply relaying a difficult concept (teachers know when that lightbulb goes off in a student's head) and emotionalizing that concept (that's a whole different look behind their eyes, and probably why teachers become teachers). How do we get that same reaction? It's probably as much for me as for them that I want them to see computing as a craft for the greater good. I have to teach my students counting, so I will forego puppies and have them design a tamperproof voting system. When they learn two-dimensional arrays it will be to monitor the flow of pollution through Lake Erie. Databases? Not fish in a bowl but it will be drug interactions. My good friend Devika Subramanian at Rice University taught me how to use disaster evacuation planning to teach optimal paths and routing instead of using chess. I can't find any of these in a textbook that teaches CS1, so I'll have to invent them.

I know that writing a textbook is difficult. But so is teaching, and so is learning. C

b See http://www.sociallyrelevantcomputing.org/images/pic17.jpg.

Further Reading

1. Buckley, M. et al. Benefits of using socially relevant projects in computer science and engineering education. In *Proceedings of the Special Interest Group on Computer Science Education Conference*, 2004; http://www.sociallyrelevantcomputing.org/SIGCSE2004SociallyRelevantProjects.pdf.
2. Buckley, M., Schindler, K., Kershner, H., and Alphonce, C. Using socially relevant projects in a capstone design course in computer engineering. In *Proceedings of the American Society for Engineering Education Annual Conference*, 2004; http://portal.acm.org/citation.cfm?id=1028174.971463.
3. Nordlinger, N., Subramanian, D., and Buckley, M. Socially relevant computing. In *Proceedings of the Special Interest Group on Computer Science Education Conference*, 2008; http://www.cs.rice.edu/~devika/SIGCSEFinal.pdf.
4. Schindler, K., Buckley, M., Kershner, H., and Alphonce, C. Partnering with social service organizations to develop socially relevant projects in computer science and engineering. In *Proceedings of the International Conference on Engineering Education*, 2004; http://www.sociallyrelevantcomputing.org/ICEE2004.pdf.

Michael Buckley (mikeb@cse.buffalo.edu) is the director of the Center for Socially Relevant Computing at the University of Buffalo, NY.