

ICER West White Paper

though more of a testimonial/report from the trenches?

Jim Fix, Reed College

I teach at a small liberal arts college. The college places strong emphasis on students pursuing, with focus and depth, a major discipline. There are interdisciplinary majors and programs of study, but they are formal rather than ad hoc. Students gain breadth in their education by a distribution requirement— students must take a two semester sequence in five areas of intellectual inquiry such as the Natural Sciences, Literature, Social Sciences. Thus, our students are asked to know one discipline well, but the distribution requirement, and the college's size, fosters a spirit of interdisciplinary curiosity, especially among the students.

Unfortunately, neither computation nor computer engineering are programs of study at this moment. My courses are offered by the Mathematics Department, but my *Introduction to Computation* course regularly has its share of English literature, History, and Religion majors, along with the more aligned Psychology, Linguistics, Biology, Chemistry, and Philosophy majors, and the naturally interested Mathematics and Physics students.

I have found that his environment has given me a particular insight into the challenge that perhaps the fields of Computer Science and Computer Engineering are facing from the outside. I've found students and colleagues to be naturally skeptical about the place of computation and computing amongst competing disciplines for study, though curious about the possible impact of technology on society. The post-dot-com boom has contributed to this harsh climate.

CS outreach to support and promote CS & technology

Because of my situation, perhaps my immediate goals make the strategies I apply seem less applicable to creating the technological workforce, since I do not support an engineering curriculum. I will leave such programs to determine their own strategies as they have more clear insights into their challenges. However, I feel my ideas and experiences have application to the development of an outreach curriculum, of creating computer science courses in secondary and undergraduate education for those who may not directly develop technology.

After all, a successful technological workforce requires support and understanding from non-technologists. Its success might be reliant, for example, on more informed policymakers. We might want extremely valuable contributions from people outside the discipline. Technological innovation is fostered and the field itself benefits from its ability to justify and explain its practice if it results in more participants.

In order for us to meet our teaching challenges, we must assert our position as an intellectual discipline. Computation is distinct, and on par with mathematics and the natural sciences. For all people, regardless of their trade or discipline, to understand and reason well about technology and technological innovation, they need to be exposed to the mathematical, natural, and the *artificial* sciences. Asserting our field's role and importance in this will be key to their success. Computer science can have a stronger impact in a younger student's general education. It's not just programming. It's not simply a road to employment.

Winning hearts and minds with a core CS intro

So, if *winning hearts and minds* is our challenge, how do we address it? How can a computer science curriculum appeal to students that likely will not be immediately employed to design and develop computer technology?

One way that I *don't* do this successfully is by asserting to my colleagues and students that computation has impact on their fields of interest. They already think they have a sense of its impact, and they are probably right, as far as they are concerned. Stressing the interdisciplinary nature of the field is only useful in execution, leading by doing, rather than in my rhetoric. I look for opportunities to collaborate when I can, I point out relationships and applications to other fields, but I still only focus on the core ideas.

I *do* find my students to be naturally interested in interdisciplinary study. They are naturally looking for the connections between the fields, their minds tie together the thinking and ideas from each of the courses they're taking. All their courses, however, demand their attention. This creates, of course, a natural competitive environment at a fine granularity. I have one chance at a student in one semester.

One strategy I suggest for a certain kind of introductory course is one that **makes its topics and ideas stand on their own** as intellectual enterprise equal to those in Mathematics and Sciences, without

emphasis on their use in the construction of technology. I *do* choose topics that are distinctly computational in nature. It is here that I find *core* computer science to be a natural winner. (It certainly is a field that won me over.)

There are a number of great stories to be told, technical ones, about the development of computing, especially if you examine its relation to mathematics, philosophy, and logic. In my introductory course, I teach computer programming, it is important to me to provide a laboratory setting for such problem solving. Students are naturally curious about how computers work, and I am naturally interested in sharing with them the general principles of computer systems' construction. More importantly, my introductory course is *driven* by my coverage of formal language and automata theory. I introduce automata and regular languages, push through context free languages and stack-based machines, and end with Turing machines, the universality of Turing-complete computational models, and finally, the halting problem. I use Sipser's text *Introduction to the Theory of Computation*. Regardless of major, I find that the students really eat these computer theory topics up. They love the problems they are asked to solve. The computational structures seem so novel, but natural, to them.

Think again about the contributions to human thought made by pursuit of the questions that led to the theory of automata: What is computation? What does it mean to automatically deduce, to automate logic? What are our limitations in this regard? How can certain structures, like natural language, be modelled?

I find these questions absolutely fascinating, and so do my students. They are distinctly computational questions, though they certainly overlap in a rich way with other areas of intellectual inquiry. Nonetheless, I want to emphasize this point: **these questions are *distinctly* computational, independent of computing software and hardware. They are the seeds that created our field, they need to be disseminated, and they're not easily forgotten.**

The possibility of interdisciplinary undergraduate CS research

I'll end with further comments on the possibilities for interdisciplinary computer science work through undergraduate research projects, a requirement of students at my college that results in a written thesis. Its success comes from insisting upon student's practice of devising an independent plan of study, collecting research materials, meeting and working closely with a faculty advisor, regularly communicating their ideas, orally and in writing, to those in- and outside their major, and producing a final product. The benefits are enormous: we've all experienced them in our graduate training.

It is possible to have students do interdisciplinary investigations in this kind of undergraduate research project. *They* are the leads in their project, and they can gather insights from a secondary advisor from outside computer science. The field of CS is, in principle, ripe with such collaborative opportunities.

I've had mixed experiences, and mixed success, with such theses, for example: on algorithms for mathematical models of natural language, on protein folding, on voting systems. In every case, it was difficult to make the work seamlessly interdisciplinary, in fact, we insured the main contribution of the work was to computer science... played it safe. However, because the culture of undergraduate research is in place at the school, students readily have opportunity to talk comfortably about their thesis work to faculty in the "other" discipline. As faculty, we serve others' students so that they will serve ours, in turn. In addition, the interdisciplinary relationship arises as a result of the student having taken coursework from the "other". I've worked with numerous students on the computational aspects of their research projects (in other disciplines) simply because they took and enjoyed my courses.

I think this leads to my original point: **create a culture of exchange between disciplines, make your (course)work appeal to an outside audience, and interdisciplinary opportunities can arise in a natural way.**

Thanks.