

ICER White Paper
Kim Bruce
Pomona College

1. Preparing undergraduates for computing careers:

The biggest challenge is getting the good students in the door. A very strange thing happened with the dot-com boom and then bust. When the boom came, the host of students more focused on making lots of money drove away some of our better students, who ended up majoring in other disciplines. When the bust came, we lost most of those there just for the money, but we had fewer of the good students left. We also lost a disproportionate number of women.

We must convince students that there is interesting work to be done in Computer Science, and there is money to support them after they graduate.

2. Transforming the educational experience:

There are a number of things that must be done. Here are a few:

- i. Let people know that CS is important, interesting, and needs a lot of new, good people. Math has done an excellent job of generating materials that let students know about all of the interesting things that could be done with math (half of which are really CS-related). While a few people have been going around telling this story (Dave Patterson, Bill Gates), much more can and should be done.
- ii. We need to stop teaching intro courses as a way of filtering out the unworthy. Several years ago mathematicians came out with a slogan that calculus should be a "pump, not a filter". We need to put a similar focus on helping students succeed in introductory courses.
- iii. Introductory courses have to be rethought. We need to figure out ways of presenting interesting and challenging material to students early in their study of computer science.
- iv. A large amount of professional programming in the future (as is true now) will be done by those who know the application area well. We must respond to this in two ways. First, we must provide opportunities for students in other areas to obtain a minor in computer science that will provide them with the knowledge and skills necessary to use computing effectively in their area of major interest. Second, we must encourage and provide space for our own majors to develop expertise in outside areas so that they can contribute productively on teams working on projects centered on those other domains.

This spring I will be teaching a course primarily aimed at students who are advanced computer science or linguistics majors. After teaching them some fundamentals, I will be attempting to get them to work in cross-disciplinary teams to investigate interesting questions in linguistics that would benefit from logical and computational tools. I'm not sure yet how to get this to work, but solving this kind of pedagogical challenge is essential.

3. Models for transforming computing education:

I believe that the 2004 Liberal Arts CS curriculum guidelines (<http://www.lacs.edu/model-curriculum.pdf>) provide a nice balance between traditional CS and those topics of increasing importance, at least for those departments operating in a liberal arts context.

I would like to see a curriculum that supports a solid, though relatively compact, common core for all CS majors and then allows them the freedom to move off in different directions. I do not believe that the common core should have every student cover a little bit of everything, but instead cover fewer but really core topics. I believe that the purpose of an undergraduate degree should be to provide a students with the foundations for later learning, as there will be a constant flow of new ideas and information to be learned and used. A master's degree should be the professional degree that prepares students with the skills for a particular job.

We need to develop ways of teaching our students how to work with those in other disciplines and ways of teaching those in other disciplines the computing skills necessary. I'd like to see us figure out a way to teach truly interdisciplinary courses to advanced students in CS and other disciplines together that would help them develop ways of pooling their different backgrounds and skills to solve problems.

4. Inhibitors and strategies:

The hardest step will be getting faculty to consider new ways of teaching introductory courses. I suspect that a careful comparison of introductory syllabi today with those from 25 years ago will show too few differences aside from programming languages.

In order to help faculty make the necessary changes we need to develop, identify, highlight, and disseminate detailed curricula for introductory courses that have been shown to be successful. There will not be a single model, but instead several models should be identified. These should be supported with detailed lecture notes, laboratories, etc., wherever possible. Of course there are huge political problems in doing this as authors and publishers will want their own books selected, but it might be possible to provide enough detail on different approaches to be of great help to others wishing to adopt a similar approach, yet not so much that it requires a single text.

A major inhibitor to getting students involved in interdisciplinary work is that most faculty in other sciences tend to resist having their students get involved. They would rather see the students do more work in their own discipline than go over and take courses in computer science. I have been surprised and disappointed at a number of offerings in computational biology for undergraduates that have drawn very few biology majors. We need to figure out a way of getting faculty in other disciplines to support the study of computer science.

5. Who might participate:

Key stakeholders are faculty, college administrators, foundations, and industry. Individual faculty will need to be enticed to participate in educational research and development to design more interesting introductory courses. Computer scientists in industry can be the most effective in developing materials that explain to students what computer scientists do in the workplace, and why the computing skills are valuable. Foundations can provide funding to support development and distribution of these materials.