

Where to Begin?

Thomas Murtagh¹
Department of Computer Science
Williams College
Williamstown, MA 01267
tom@cs.williams.edu

The majority of introductory computer science courses for potential majors focus on the development of programming skills. This is an obvious place to begin. Programming skills are essential for more advanced study. In many ways, however, it is a terrible way to introduce students to computing. Most students taking CS 1 enter the course with relatively little knowledge of the nature of our discipline. Any plans they may have to concentrate in computer science are based on flimsy evidence. Experiences from surfing the web to playing video games may have convinced them that they like computers, but this does not mean they will like computer science. Recent fluctuations in enrollments certainly suggest that interest in computer science is often based more on the job market than on knowledge of our field.

The earlier students can refine the vague academic goals with which they typically begin their undergraduate careers into clear plans based on concrete knowledge of the disciplines they choose to pursue, the more fulfilling their educations will be. Compared to many other disciplines, students receive very little information about computer science in elementary and secondary schools. Accordingly, we have a responsibility to provide more in CS 1 than prerequisites for those who decide to concentrate in computer science. It is important that we provide students with a view of the nature of our field sufficiently detailed to enable them to make informed decisions on whether to continue their studies in computer science.

Our students would not be the only ones to benefit if introductory computer science courses conveyed a broader picture of the field. As a discipline, we would benefit greatly if the students who decided to pursue more advanced studies really knew what they were undertaking. I have frequently been disappointed when bright students have abandoned computer science after one or two semesters because they want more than “just programming,” and depressed by students in upper-level courses who complain that they “hate math.” In addition, an introduction that presented our discipline in a context that highlighted ways in which computer science has impact on daily life in our society might help attract a set of students to our programs that was more diverse.

At first glance, this may just sound like an argument for the breadth-first approach. This approach, clearly elucidated over fifteen years ago in the “Computing as a Discipline” report, suggests that introductory CS courses should provide a breadth-first introduction to our discipline[DCG⁺89]. Many approaches to the design and implementation of breadth-first courses have been proposed. Unfortunately, the ACM/IEEE Computing Curriculum 2001 report concludes “We have not been able to identify any such models that meet our acceptance criterion of successful implementation by faculty other than the originator.” [CC01] Strikingly, the report still encourages the development of such courses despite its gloomy conclusion that no such course has yet been successful. It even hints at what might be needed:

The many disparate topics typically found in a breadth-first course must be tied together into an integrated whole. Students must not see the course as a collection of interesting but unrelated topics in a[n] “if this is Tuesday it must be computer organization” style.

The challenge, then, is to devise courses that present a tightly integrated set of topics while simultaneously providing a broad and diverse introduction to the nature of our discipline.

Perhaps the way to overcome this challenge is to focus on a “tightly integrated set of topics” rather than on the “broad and diverse introduction.” To make this notion concrete, consider an example from another discipline. The Sociology department at the University of Massachusetts allows students to start with any of five different “first” courses. One of these has a title analogous to a typical CS 1 course, “Sociology 110 — General Introduction to Sociology.” The other four courses have titles like “Self, Society and Interpersonal Relations” and “Race, Gender, Class, and Ethnicity.” The UMass sociology department accepts all of these

¹This work was supported in part by NSF DUE-0442954.

as introductory courses for potential majors. Even though most of the course titles don't say anything about introducing students to the discipline of sociology, the faculty of the department apparently think that they provide an appropriate introduction to their field. How can this be?

The key is to appreciate that we can enable students to understand the nature of our discipline while focusing on just one subfield or problem area within computer science. We can accomplish this by using the area on which the course focuses to expose students to the types of problems our discipline addresses, the techniques we use to attack these problems, and the forms taken by the solutions we devise. At a minimum, we would expect this to include examples of:

- precise specification of algorithms for complex procedures in both pseudo-code and executable code,
- explicit evaluation of schemes for representing information,
- program/system decomposition and layering,
- mathematical analysis to predict performance, and
- rigorous arguments to ensure correctness.

This semester, I am offering a course based on this approach for the first time. Unlike the UMass sociology department, our size does not allow us to offer multiple introductory courses, so the new course, Digital Communications and Computation, is the only CS 1 course we are offering this semester. The course is focused on problems encountered in the study of computer networks. It covers both programming basics and topics ranging from data compression techniques to routing algorithms. We can certainly imagine similar courses focused on many other topics within computing.

There are many obstacles to wide adoption of this approach. Inertia is a clear impediment. Such courses are likely to produce students with less programming preparation than the typical introductory course but with more mathematical preparation. This implies that a transition to this type of introductory course will result in changes that percolate through the more advanced courses in the curriculum. At the very least, topics will have to be shifted about. In addition, integration in the first course may change student expectations of later courses. Will such students tolerate a "just programming" approach in their second course?

Lack of curricular materials is another concern. Integrating the programming component of an introductory course with another topic that serves as the focus of the course is likely to demand an approach to teaching programming that is not supported by current texts. In our course, for example, students are creating network connections by the second lab. What existing book introduces networking primitives so early? At the same time, the texts available for a topic like computer networking do not present the material at a level appropriate for this type of introductory course.

Finally, designing such courses would be highly demanding even if the availability of materials was not an issue. The design must be driven by two competing constraints: the need to provide a credible coverage of the topic that is the explicit focus of the course, and the underlying goal of providing a broad overview of the nature of computer science as a discipline.

Despite these obstacles, our initial experience teaching such a course at Williams has been quite promising. We hope that our experience with this approach over the next few semesters will provide insights into how we can more effectively introduce our discipline to the nation's undergraduates.

References

- [DCG⁺89] Peter J. Denning, D. E. Comer, David Gries, Michael C. Mulder, Allen Tucker, A. Joe Turner, and Paul R. Young. Computing as a discipline. *Commun. ACM*, 32(1):9–23, 1989.
- [CC01] ACM/IEEE Joint Task Force on Computing Curricula. Computing curricula 2001. *J. Educ. Resour. Comput.*, 1(3es):1, 2001.