12-2011

Is Computer Science a Relevant Academic Discipline for the 21st Century

Doug Baldwin
SUNY Geneseo, baldwin@geneseo.edu

Is Computer Science a Relevant Academic Discipline for the 21st Century?

Doug Baldwin

Department of Computer Science

SUNY Geneseo

At least in the United States, the answer to the title question seems to be "no." Far from being seen as a "discipline," i.e., an area of research and study with a distinctive body of knowledge and methods of inquiry, computing in general is now seen as body of technology (both hardware and software) to be applied in other areas. This view is coming to define "computing," sweeping up students, educators, and industry leaders on its way. What this view of computing as technology overlooks, however, are computing's theoretical and scientific foundations in computer science.* The longer we neglect these foundations and the deeper we subordinate them to other interests, the weaker the entire computing enterprise becomes.

Until about the year 2000, "computer science" as an academic discipline studied most things related to computing. "Computer engineering" concerned itself with hardware aspects of computing, and "software engineering" with the effective production of software, but by and large computing was taught and studied by departments of computer science. In the decade from 2000 to 2010 this model disintegrated. Undergraduate and secondary enrollments in computer science dropped. Many colleges and universities responded by creating programs in "information technology," "information science," or "information systems." Interdisciplinary programs with computing components, such as bioinformatics, game design or web design, appeared. Undergraduate software engineering programs proliferated. In all cases, the hope was that the more applied aspects of computing would appeal to students even if traditional computer science did not. Computer science programs themselves began to place more emphasis on computing's applications. At the secondary level, high schools, in which financial

---

* This essay distinguishes "computing" and "computer science," recognizing that "computer science" is only one of many computing fields today. I use the term "computer science" to mean an area of study or research concerned with computing broadly, but particularly addressing its theoretical foundations.

pressures were mounting and computer science was generally an elective, were only too happy to eliminate computer science offerings outright; a handful of colleges followed suit.

Today, a norm in which the study of computing is dispersed into application areas appears to be emerging, and stakeholders, for the most part, seem content with it. In this context, "application areas" denotes a wide variety of disciplines concerned with creating or managing software or hardware applications, ranging from software and computer engineering to the various "information" fields to traditionally non-computing fields that now have computational branches (e.g., computational sciences, digital humanities, etc.)

Enrollments in applied computing disciplines are strong now, even while enrollments in computer science rebound. For instance, the 2009-10 Taulbee survey, which now surveys Ph.D.-granting departments in "information" fields (so-called "I" departments) as well as computer science and computer engineering, finds significant numbers of students in the I departments particularly at the bachelors and masters levels: just under 1/6 of bachelors degree recipients, and about 1/5 of masters degrees. While the survey's authors caution that I school data is too new to draw statistical conclusions from, the numbers are substantial enough to suggest that these programs are not mere passing fads.

In college and university computer science departments, applications of computer science have a new prominence. For example, "media computation," an introduction to programming in the context of its application to image and sound manipulation, has spread to a wide variety of colleges, universities, and high schools (see http://coweb.cc.gatech.edu/mediaComp-teach/37 for some examples). Some computer science programs require "applied' computer science subjects (e.g., numerical methods, computational science, computer graphics, artificial intelligence, robotics, etc.) as core parts of their majors (http://www.cs.dartmouth.edu/site-content/site/a-major-redesign.php is a particularly clear example). Research interests featured on department Web pages frequently include problems motivated by, or results of interest to, other disciplines (biology, biochemistry, and medicine seem particularly common); my own research addresses problems in computer graphics motivated by visualizations for particle physics.

At the high school level, computing seems firmly set as a supporting skill for the traditional sciences and mathematics. This is exactly how the recent NRC "Framework for K-12 Science Education" addresses computing (see http://www.nap.edu/catalog.php?record_id=13165), and the Common Core State Standards for Mathematics (http://www.corestandards.org/assets/CCSSI_Math%20Standards.pdf) make frequent

mention of computer algebra systems and similar tools for understanding or visualizing mathematical ideas, but no mention of learning computer science or computational thinking. Despite influential countervailing voices, notably advocacy efforts by the Computer Science Teachers' Association and ACM, and a report on STEM education from the President's Council of Advisors on Science and Technology ("Prepare and Inspire: K-12 Education in Science, Technology, Engineering, and Math (STEM) for America's Future") computer science is on track to become a service discipline in America's secondary curriculum.

Does it matter if computer science disperses over a myriad of applied computing fields and disciplines that draw on computing for their own ends? One of the triumphs of computing is that it has transformed nearly every other area of human activity, and to some extent this dispersal is just a logical consequence of that transformation.

However, if time amplifies the tendency to see computing *only* as a supporting service for other disciplines, as seems to be happening in K-12 standards, the results will be catastrophic, for several reasons:

1. Neglected topics. Significant computing ideas can be and have been developed in other disciplines, but some fundamental areas of computer science have no call on those disciplines' attention. For example, past work on basic theories of what it means to compute has led to powerful and widely used tools—regular expressions, parsers for programming and other languages, etc. There are still open questions in this area, for instance whether fast factoring algorithms exist or what the potential of quantum computing is, whose answers, if found, will impact applications in security and many other areas. Yet people working on day-to-day problems in these areas are unlikely to have the inclination, time, or theoretical background to work on those questions. Similar arguments could be made about programming language semantics and applications concerned with parallel computing, security, etc. Neglect is a concern in education as much as in research: students who aren't exposed to certain areas of computing will eventually become professionals who don't appreciate the value of those areas, if they know the areas exist at all.
2. Isolated sub-disciplines. As computing fragments into application areas, computing education and research will concentrate in those areas' curricula and publications. While each area can appropriately teach its distinctive problems and methods, it is unnecessarily duplicative for each to teach common foundations in programming, basic algorithms, or standard data representations. Further, students in fields that don't teach computing application courses nonetheless benefit from a general exposure to computational thinking, but it is unclear where they will get this exposure if computing comes to be taught only in application curricula (e.g.,

should a philosophy major learn computational thinking in a computational science course? in a business information systems course? perhaps in a communication arts Web design course?) Common foundations also mean that research results from one application area are often relevant to others, but sharing of such results is difficult if the areas don't have publications in common (although scholarly search services such as Google Scholar may to some extent mitigate this problem).

Computing's fragmentation is well under way, and is an unavoidable consequence of its maturation. However, fragmentation does not have to mean a collection of technology applications with no core science. The emerging computing disciplines need to agree what each does and does not cover, and what common scientific foundation they rest on. More importantly, they need to reach out to computational sub-disciplines in the other sciences, business, humanities, and elsewhere to help them see that their applications also rest on the same foundation. Similarly, the computing community needs to educate policy makers and K-12 standards setters about the relationship between science and applications in computing. If these things happen successfully, computer science can stand in the same relationship to the applied computing areas as the more traditional sciences stand to their applied science and engineering fields. Failure, on the other hand, will leave computing a collection of sterile disciplines unable in the long run to deliver on the social and economic promises they offer.