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

Douglas Baldwin, *SUNY at Geneseo*

The current view of computing as technology overlooks the discipline's theoretical and scientific foundations in computer science, weakening the entire computing enterprise.

At least in the US, the answer to the question posed in the title seems to be no.

Far from being seen as a discipline—an area of research and study with a distinctive body of knowledge and methods of inquiry—computing in general is now regarded as body of technology (both hardware and software) to be applied in other areas. This view is coming to define what is meant by "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--an area of study or research that is concerned with computing in general, but particularly addresses its theoretical foundations, and thereby is distinguished from more applied computing fields. The longer we neglect these foundations and the deeper we subordinate them to other interests, the weaker the entire computing enterprise becomes.

AN EVOLVING MODEL

Until about 2000, CS as an academic discipline studied most things related to computing. Computer engineering concerned itself with the hardware aspects of computing, and software engineering with the effective production of software, but, by and large, computing was taught and studied in CS departments.

In the decade from 2000 to 2010, this model disintegrated as undergraduate and secondary CS enrollments declined. 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, were introduced. 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 CS didn't. CS programs themselves began to place more emphasis on computing applications. At the secondary level, high schools, in which financial pressures were mounting and CS was generally an elective, were only too happy to eliminate these offerings outright; a handful of colleges followed suit.

In what is now becoming the norm, the study of computing is dispersed into application areas, and stakeholders, for the most part, seem content with this. In this context, "application areas" denotes a wide variety of disciplines concerned with creating or managing software or hardware, ranging from software and computer engineering to the various information fields to traditionally noncomputing fields that now have computational branches such as computational sciences, digital humanities, and so on.

Enrollments in applied computing disciplines are strong now, even while CS enrollments rebound. For example, the 2009-2010 CRA Taulbee Survey, which now includes PhD-granting departments in information fields (so-called "I departments") as well as CS and computer engineering, found significant numbers of students in these departments, particularly at the bachelor's (just under one-sixth) and master's (about one-fifth) levels. While the survey's authors caution that I-department data is too new to draw statistical conclusions from, the numbers are substantial enough to suggest that these programs aren't mere passing fads.

In college and university CS departments, applications of CS have a new prominence. For example, media computation, an introduction to programming in the context of image and sound manipulation, has spread to a wide variety of colleges, universities, and high schools (http://coweb.cc.gatech.edu/mediaComp-teach/37). Some CS programs require applied CS subjects—for example, numerical methods, computational science, computer graphics, artificial intelligence, and robotics—as core parts of their majors (www.cs.dartmouth.edu/site-content/site/a-major-redesign.php). Research interests featured on CS department webpages frequently include problems motivated by 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 how the recent National Research Council's "Framework for K-12 Science Education" addresses computing (www.nap.edu/catalog.php?record_id=13165). The Common Core State Standards for Mathematics (www.corestandards.org/assets/CCSSI_Math%20Standards.pdf) make frequent references to computer algebra systems and similar tools for understanding or visualizing mathematical ideas, but they don't mention learning CS 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 titled "Prepare and Inspire: K-12 Education in Science, Technology, Engineering, and Math (STEM) for America's Future" (www.whitehouse.gov/sites/default/files/microsites/ostp/pcast-stemed-embargoed2.pdf), CS is on track to become a service discipline in America's secondary curriculum.

Students in fields that don't teach computing application courses nonetheless benefit from a general exposure to computational thinking.

A RISKY DISPERSAL

Does it matter if CS disperses over myriad applied computing fields and disciplines that draw on computing for their own ends? One triumph 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.

Neglected topics

Significant computing ideas have been developed in other disciplines, but some fundamental CS areas don't receive attention from those disciplines. For example, past work on basic theories of what it means to compute has led to powerful and widely used tools, including regular expressions, parsers for programming languages (and other languages), and so on.

There are still open questions in these areas, for example whether fast factoring algorithms exist or what the potential of quantum computing is. If found, the answers will impact applications in security and many other areas. Yet people working on day-to-day problems in these impacted areas are unlikely to have the inclination, time, or mathematics background to work on answering these theoretical questions. Similar arguments could be made about programming language semantics and applications concerned with parallel computing, security, and so on.

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 even know the areas exist.

Isolated subdisciplines

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's 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. However, it's unclear where they'll get this exposure if computing is eventually taught only in application curricula. Should, for example, a philosophy major learn computational thinking in a computational science course, a business information systems course, or perhaps a communication arts Web design course?

Common foundations also mean that research results from one application area are often relevant to others, but sharing such results is difficult if the areas don't have publications in common (although services such as Google Scholar may mitigate this problem to some extent).

Computing's fragmentation is well under way, and is an unavoidable consequence of its maturation. However, fragmentation doesn't have to mean a collection of technology applications with no core science.

The emerging computing disciplines need to agree on what each does and doesn't cover, and what common scientific foundation they rest on. More importantly, they need to reach out to computational subdisciplines 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 policymakers and K-12 standards setters about the relationship between science and applications in computing.

If these things happen successfully, CS can stand in the same relationship to the applied computing areas as the more traditional sciences stand with their applied science and engineering fields. Failure, on the other hand, will leave computing a collection of sterile disciplines unable to deliver in the long run on the social and economic promises they offer. C

Douglas Baldwin *is a professor of computer science at SUNY at Geneseo. Contact him at baldwin@geneseo.edu.*

Editor: Ann E.K. Sobel, Department of Computer Science and Software Engineering, Miami University; sobelae@muohio.edu