David A. Patterson

Computer Science Education in the 21st Century

To draw students to CS, we must first look to create a curriculum that reflects the exciting opportunities and challenges of IT today versus the 1970s. Future students and faculty would greatly benefit from a reinvigorated CS curriculum.

This letter is based on my position statement for a workshop on the preparation of IT graduates for 2010 and beyond [4].

Whereas in the past we created obstacles to reduce the number of CS majors, today we must recruit students to have the work force needed to meet the challenges and opportunities of information technology in this century. We should take advantage of the reduced pressures from the dip in enrollments to revamp our curriculum.

First, let's start with the state of the world in 2006 rather than 1976. Second, let's create courses that we would love to take if we were students, and that we would love to teach if given the chance. Such enthusiasm would be attractive and contagious.

Rather than wax on philosophically, I'll confine my remarks to four concrete suggestions: two technological upgrades and two examples of courses I would love to take and teach. All these suggestions leverage technology not available when the CS curriculum was first created.

Technological Upgrade #1 for 21st Century CS: *Use Tools and Libraries.*

There is a huge disconnect between the experience of most professors, who have never worked as professional programmers and often write software for a 30-year-old environment, and the way in which cutting-edge software is written today.

For example, although many professors use a more recent programming language like Java, surprisingly few have embraced modern programming environments like the Eclipse development platform and the JUnit testing framework. Moreover, a great many exciting programming projects today build on existing components such as .NET or NETBeans.

For many CS courses, a dramatic change would simply be if students first wrote a clear specification and then built software using modern tools and software components.

Technological Upgrade #2 for 21st Century CS: *Parallelism.*

In case you weren't paying attention, the era of doubling performance every 18 months ended in 2002, as the figure here documents [3]. Three factors have combined to grind uniprocessor performance improvement almost to a halt.

- The lack of additional power for a chip to dissipate,
- The lack of additional instruction-level parallelism to exploit, and
- The lack of improvement in memory latency.

This sea change is demonstrated by the change in direction of microprocessor companies away from increasing clock rates and uniprocessor performance. AMD, Intel, IBM, Sun Microsystems, among others