



Peter J. Denning

Computer Science and Software Engineering: Filing for Divorce?

Recent proposals to license software engineers have strained the uneasy tension between computer scientists and software engineers. Software engineers tend to believe that certification is valuable and licensing is inevitable; they want significant changes in the curriculum for professional software engineers. Many computer scientists tend to believe certification is unnecessary and licensing would be harmful because it would lock in minimal standards in a changing field of rising standards. Frustrated, a growing number of software engineers want to split off from computer science and form their own academic departments and degree programs. Noting other dualities such as chemical engineering and chemistry, they ask, why not software engineering and computer science? (Sept. 1997, "Risks").

No such rift existed in the 1940s and 1950s, when electrical engineers and mathematicians worked cheek by jowl to build the first computers. In those days, most of the mathematicians were concerned with correct execution of algorithms in scientific application domains. A few were concerned with models to define precisely the design principles and to forecast system behavior.

By the 1960s, these engineers and programmers were ready for marriage, which they consummated and called computer science. But it was not an easy union. Academic computer scientists, who sought respect from traditional scientists and engineers for their discipline, loathed a lack of rigor in application programming and feared a software crisis. Professional programmers found little in computer science to help them make practical software dependable and easy to use. Software engineers emerged as the bridge builders, responding to the needs of professional programming by adapting computer science principles and engineering design practice to the construction of software systems.

But the software engineers and computer scientists did not separate or divorce. They needed each other. Technologies and applications were changing too fast. Unless they communicated and worked together, they could make no progress. Their willingness to experiment helped them bridge a communication gap: Software engineers validated new programming theories and computer scientists validated new design principles.

But that was a long time ago. Hasn't the field matured enough to permit the two sides to follow separate paths successfully? I think not. The pace of technological change

has accelerated. Even in the traditional technologies such as CPU, memory, networks, graphics, multimedia, and speech, capacity seems to double approximately every 18 months while costs decline. Each doubling opens new fields that form at interdisciplinary boundaries. Some examples:

- New computing paradigms with biology and physics including DNA, analog silicon, nanodevices, organic devices, and quantum devices
- Internet computations mobilizing hundreds of thousands of computers
- Neuroscience, cognitive science, psychology, and brain models
- Large scale computational models for cosmic structure, ocean movements, global climate, long-range weather, materials properties, flying aircraft, structural analysis, and economics
- New theories of physical phenomena by "mining" patterns from very large (multiple) datasets

It is even more important today than in the past to keep open the lines of communication among computer scientists, software engineers, and applications practitioners. Even if they do not like each other, they can work together from a common interest in innovation, progress, and solution of major problems. The practices of experimentation are crucial in the communication process. A recent study suggests that such practices could be significantly improved: Zerkowicz and Wallace found that fewer than 20% of 600 articles advocating new software technologies offered any kind of credible experimental evidence in support of their claims (see Zerkowicz and Wallace as well as Tichy in *IEEE Computer*, May 1998).

Separation between the theory and engineering has succeeded in other disciplines because they have matured to the point where they communicate well among their science, engineering, and applications branches. A similar separation would be a disaster for computer science. Spinning off software engineers would cause communication between engineers, theorists, and application specialists to stop. Communication, not divorce, is the answer. **□**

PETER DENNING (pjd@gmu.edu) was a former president of ACM and recently chaired the Publications Board while it developed the ACM digital library.