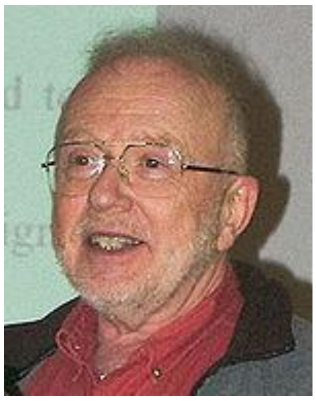
Software Engineering

ACM Fellow David Parnas, 2004



What advice do you have for computer science/software engineering students?

Most students who are studying computer science really want to study software engineering but they don't have that choice. There are very few programs that are designed as engineering programs but specialize in software.

I would advise students to pay more attention to the fundamental ideas rather than the latest technology. The technology will be out-of-date before they graduate. Fundamental ideas never get out of date. However, what worries me about what I just said is that some people would think of Turing machines and Goedel's theorem as fundamentals. I think those things are fundamental but they are also nearly irrelevant. I think there are fundamental design principles, for example structured programming principles, the good ideas in "Object Oriented" programming, etc.

What is the most often-overlooked risk in software engineering?

Incompetent programmers. There are estimates that the number of programmers needed in the U.S. exceeds 200,000. This is entirely misleading. It is not a quantity problem; we have a quality problem. One bad programmer can easily create two new jobs a year. Hiring more bad programmers will just increase our perceived need for them. If we had more good programmers, and could easily identify them, we would need fewer, not more.

What is the most-repeated mistake in software engineering?

People tend to underestimate the difficulty of the task. Overconfidence explains most of the poor software that I see. Doing it right is hard work. Shortcuts lead you in the wrong direction and they often lead to disaster.

What are the most exciting/promising software engineering ideas or techniques on the horizon?

I don't think that the most promising ideas are on the horizon. They are already here and have been here for years but are not being used properly. A few years ago, I met an experienced software development manager who had just uncovered a memo I wrote for his company in 1969.

He told me, "If we were now doing what you told us then, we would be far ahead of where we are now." The biggest payoff will not come from new research but from putting old ideas into practice and teaching people how to apply them properly. There is much more research to do and we have much to learn, but the priority should be put on technology transfer and education.

Source: ACM Special Interest Group on Software Engineering (SIGSOFT) Software Engineering Notes "ACM Fellow Profile: David Lorge Parnas" Retrieved 29 February 2004 from http://www.acm.org/sigsoft/SEN/parnas.html