There is still a war going on

The mediocre mistrust the exceptional man because they don't understand him, and they fear him because he can do things beyond their comprehension. As far as human memory goes, this circumstance has led to bloody conflicts — Christ was neither the first nor the last excentric to be crucified.

The 20th century has created new battle-fields for this old conflict in the form of the high-technology industries; there, the battles are between the managers/beancounters on the one hand, and the scientists/technologists on the other. As a service to my younger readers I point out that these battles are fierce and that no quarter is given. We regularly have to be reminded of these battles because the scientists have a tendency to lose them: firstly, they are in a minority, and secondly, their primary interest is their science and not these battles, this in contrast to the manager, who considers this battle his main challenge.

The battle-cries of the managers are absolutely standard, they have not changed since WW II. The scientist is attacked as

individual — unruly, rugged individualists being much harder to control than committees — by organizational insistence on teamwork. The scientist is attacked in his professional competence by organizational insistence on interdisciplinary research. Finally, the scientist's right of existence is denied by the postulate that there is nothing really left to be discovered. [We hear all these slogans now, but they are carefully described in [0] d.d. 1956.]

It seems that, these days, the managers are winning the battle: even high-technology companies are now shedding their scientists and engineers at a high rate. [I had already decided to write this note when I found in the International Herald Tribune (no. 34,429 d.d. 1993.11.08, p. 13) in an article about the current internal reorganization of the Matsushita company: " "Marketing and sales people have won out over the engineering/production side," Mr. Goto said."] You can ask how these managers now intend to run their businesses, but the answer is quite simple: more like trading companies and some sort of financial institutions, these at least being activities the managers think they understand.

An unpleasant complication is that the world of finance is so disreputable and that a whole bunch of new euphemisms has to be introduced to cover up this new layer of dishonesty. For instance, there is no such thing as "making money": some people succeed in transferring money out of the pockets of others into their own. A number of years ago, the institutions that try to become rich by speculating at someone else's expense - in Western Civilization known as "bank" - needed a new euphemism to cover their activities and the started to refer to themselves as "the financial industry". Lately, they have started to refer to their more elaborate extortion schemes as "(sophisticated) banking products"! Industries are expected to produce something, aren't they?

All this is particularly grievous for the computing scientist. For 40 years, the computer industry has completely ignored the findings of computing science, and, now the computer industry is in trouble, its management gives the blame to the computing science community! We failed to do the relevant research, failed to train the work force that would solve industry's problems, etc.. But the managers themselves

are, of course, a major part of the problem, something I was relatively slow to discover.

Decades ago I realized (in Europe) that we should not confuse the intrinsic difficulties of automatic computing with the problems engendered by the shortcomings of the US educational system. Too many US Colleges of Education have operated on the principle that it is more important that their students learn to teach than that they master the material to be taught, an underestimation of the significance of the intellectual contents of proper teaching that has, for instance, led to a general abhorrence of mathematics (and the corresponding incompetence) that must be seen to be believed. (Needless to say, in the century in which the widespread availability of programmable computers has made familiarity with formal techniques more rewarding than ever, this progressive demathematization of the USA is a particularly tragic development for that country.)

Later I learned that we should neither confuse the intrinsic difficulties of automatic computing with the problems gener-

ated in industry by the prevailing managerial preconceptions. Analogous to the Colleges of Education, too many of the Schools of Business Administration operate on the principle that it is more important that their students learn how to manage than that they acquire any understanding of the process to be managed. Consequently, software managers who have not the foggiest notion of what programming is all about, are only too common. To compound the problem, many managers still strive for the traditional goal of making (in the name of stability) the company as independent as possible of particular abilities of its employees — even if this seems hard to defend in any high-technology industry; they tend to attract incompetent programmers as they would not know how to employ competent ones. (A recent CS graduate got her first job, started in earnest on a Monday morning and was given her first programming assignment. She took pencil and paper and started to analyse the problem, thereby horrifying her manager 1½ hours later because "she was not programming yet". She told him she had been taught to think first. Grudgingly the manager gave her thinking permission

for two days, warning her that on Wednesday she would have to work at her keyboard "like all the others"! I am not making this up.) And also the programming manager has found the euphemism with which to lend an air of respectability to what he does: "software engineering".

When the term was coined in 1968 by F.L. Bauer of the Technological University of Munich, I welcomed it. I was at the time at the Department of Mathematics at the Eindhoven University of Technology, our department's graduates earned the academic title of "Mathematical Engineer", and I interpreted the introduction of the term "software engineering" as an apt reflection of the fact that the design of software systems was an activity par excellence for the mathematical engineer.

But that interpretation of the term did not sail well. As soon as the term arrived in the USA, it was relieved of all its technical content. It had to be so for in its original meaning it was totally unacceptable: firstly, mathematics (and certainly formal mathematics) was widely considered impractical and irrelevant, secondly, the public at large had been brain-

washed by IBM. (1968 was also the year of the IBM ad in Datamation, showing in full colour a smiling Susie Mayer, who had solved all her programming problems by switching to PL/I !) The dilution was further eased by the fact that the Anglo-Saxon "engineer" is more vocational, is closer to the "technician" and is of much lower intellectual (and social) status than his Continental counterpart. (I did some probing on how university departments are ranked in the USA; my impression is Education and Business Administration at the bottom, with Engineering usually as a close third.)

In the mean time, software engineering has become an almost empty term, as was nicely demonstrated by Data General who overnight promoted all its programmers to the exalted rank of "software engineer"! But for the managing community it was a godsend which now covers a brew of management, budgeting, sales, advertising and other forms of applied psychology.

Ours is the task to remember (and to remind) that, in what is now called "software engineering", not a single sound engineering

principle is involved. (On the contrary: its spokesmen take the trouble of arguing the irrelevance of the engineering principles known.) Software Engineering as it is today is just humbug; from an academic — i.e. scientific and educational — point of view it is a sham, a fraud.

Universities are always faced with something of a problem when there is a marked discrepancy between what society asks for and what society needs. In our case the need is clear: the professional competence of the Mathematical Engineer, familiar with discrete systems design and knowing how to use formal techniques for preventing unmastered complexity from creeping in. But said war "out there" all but prevents this need from being perceived, and what our immediate industrial environment overwhelmingly seems to ask for is different brands of snake oil, Software Engineering, of course, being one of them. And as, with the recession lasting longer and longer, the external pressures on the Universities to do the wrong things only mount, it is only to be expected that part of the campus is going to be included in the battlefield.

The task of the first-class University, how-

ever, is absolutely clear. Industry being its customer, consultancy must tell industry what it wants to hear; it is the task of the first-class University to tell industry what it does not want to hear, whereby it is the rôle of its scientific authority to ensure that the sting of the academic gadfly really hurts.

In my experience -gathered on different continents-, it is impossible to completely prevent all snake oil peddling on campus, but by openly despising it one can at least save the integrity of the academic enterprise as a whole.

[0] Whyte, William H., "The Organization Man" Simon & Schuster Inc., New York 1956

Austin, 3 December 1993

prof. dr. Edsger W. Dijkstra
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712-1188
USA