Software Development: Fashioning the Baroque. By Darrel Ince. Oxford University Press, Oxford, United Kingdom, 1988, Price £7.95, ISBN 0-19-853757-1 (paperback); Price £17.50, ISBN 0-19-853757-3 (hardback).

Handicapped as I am by the lack of formal education in assorted arts and sciences of software, I have a nagging feeling I ought to make up for it by reading all sorts of deep books that would explain to me the principles of what I have been doing for the last thirty years or so. Mercifully, it is becoming impossible to keep abreast with the published literature of the subject; apparently, with the increasingly frequent replacement of (human) editors by wordprocessors, the publishers' natural resistance to scribblers has been paralysed. Fortunately, among innumerable bantam-weight books on software (whatever you do, don't buy the paperback version: you'd be missing the only two solid parts!), Darrel Ince's clearly stands out.

A large part of the book is devoted to author's fervent criticsm of English as a means of clear, consistent and complete statement of one's ideas. The remainder provides a splendid example of just how bad the English prose could be. By cleverly contradicting himself on every point he makes, Mr. Ince has thoroughly convinced me that for the sanity of mankind the use of prose should be licensed even more strictly than firearms are in civilised countries. (To fundamentalists, who would oppose my conclusion on the ground that the fault lies not with the use of prose but with the lack of any ideas to express, I would reply that ideas Mr. Ince has aplenty, some truly breath-taking, such as, for example, the proposed solution to various software crises by contracting the programming out to hobbyists by mail-order, particularly recommended for high-security systems.)

The overall effect is somewhat spoiled by the fact that the (very few) examples of somewhat stricter means of expression in the book all contain gross errors. On the other hand, since Mr. Ince insists that no program ever is bug-free, perhaps I am being unfair to him: by publishing bug-infested programs he is only making a point!

I also liked the way in which he ridiculed the use of wordprocessors. The book contains numerous examples of what can go wrong with it: duplicated words, meaningless substitutions hastily accepted from a list of options supplied by a spell-checker let loose on a poorly typed text, multiple occurrences of the same paragraphs transferred from a chapter to another (but not deleted in the first place)—you will find all these and then some! Occasionally, I felt the author carried it too far, however. For instance, surely it would be enough to duplicate a drawing out of context, replicating it four times seems a bit crass!

The pitfalls of keeping one's illustrations in a file separate from the text are beautifully shown by the total lack of coordination between the legend on the figures and the text that refers to them. (If the Oxford University Press has any human copy-editors left, I bet Mr. Ince had a hard time convincing them that all such unmarked exhibits were to be left intact, but succeed he did and how!)

The book is also a valuable source of information. For instance, the term "rapid prototype" used to bother me (I am easily bothered by terms everybody but I seem to understand perfectly well). Now I finally know that it is a program which (i) enjoys the triad of properties: quick, dirty and slow, (ii) is produced by hackers. Incidentally, I learned what a "hacker" is, too: he/she is a middleranking, pin-stripe executive in charge of substantial budget, with a comfortable office and at least a substantial share of a secretary. (This is the contemporary species; the hackers of 60s, dressed like bag-men, lived in cupboards, had to cope with rudimentary computing equipment, and were eventually dispersed by pin-striped quality assurance men.) Oh, if you want to meet a contemporary hacker, Mr. Ince obligingly tells you where you can find one: they tend to congregate around UNIX (no trade mark!) because it gives them the ability to develop these quick, dirty and slow programs. No, Mr. Ince does not recommend hacker-safari parties, most likely because he cannot make his mind up whether they should be exterminated, protected or bred.

Penetrating definitions are not restricted to merely software terms. For example, for quite some time I thought I knew what the "set theory" was. I did not even suspect how terribly inadequate was my snug knowledge. It turns out that, for a true-blue software expert, the set theory is that what governs the behaviour of groups of objects.

And so I could continue listing the delights of Mr. Ince's book. Nothing, however, short of reading the original, could exhibit all its charms. So I rest my case.

PS. I think I can explain (at least to my satisfaction) the reasons of all idiosyncrasies of Mr. Ince's book. All, that is, but one. Mr. Ince seems to be very much impressed by Fred Brooks' publications. He quotes from him at length, he refers to his both popular opera. Why does he then invariably misspell the name? (With at least a dozen occurrences, a plain typo is out of question; is it perhaps one more helpful feature of Mr. Ince's wordprocessor that I am not aware of? or a result of bebugging, so much admired by the author? Experts, please explain!)

W.M. Turski
Warsaw University
Warsaw, Poland