programmer corresponds to a technician, a software engineer to an electrical engineer and a computer scientist to a physicist. The real distinction lies in that the programmer is involved with the realisation of a specific computer application, in some aspect of program development or evolution. The software engineer, on the other hand, is primarily concerned with the processes, methods, tools and so on used by programmers and others in such development, in the management of a large development project where process issues dominate or in the study of the underlying theory.

The author's style is informal with a frequent expression of *opinion* and suggestions of *you could* or *you might* rather than judgement or recommendation. One must acknowledge his candor in not being decisive but this attitude is not helpful to someone studying the book in order to learn and apply.

This is an *itsy bitsy* book that contains much useful, well-described, material that does not, however, hang together to provide a coherent thesis, or even introductory text, on software engineering. One suspects that the author has not read or mixed widely in the software engineering community and is reflecting aspects of the work of a very limited number of people. This is substantiated by the very incomplete bibliography that contains no references to the various software engineering journals, to the Proceedings of the ICSE conferences or of the CASE and Process workshops for example. His list of references is woefully inadequate with names like Balzer, Boehm, Belady, Cheatham, Curtis, Hoare, Howden, Cliff Jones, Lehman, Musa, Osterweil, Riddle, Stenning, Turski, Zave, to cite just a few, totally absent. All these individuals have published contributions that must be considered basic to the current state of software engineering and even an introductory text should take cognisance of, at least, some of the concepts introduced by them.

What then does this text achieve. It lays great stress on the inevitability of *change* and the implications of this on planning and execution of development and implementation. This fundamental fact and how it may be tackled deserves wide attention and that alone makes the book a useful addition to the reference works on the subject. There is much else of interest to the beginner programmer or to the more experienced who wish to obtain a bird's eye view of aspects of software engineering technology. But it is not sufficiently wide or comprehensive to constitute a software engineering text or even to an introduction to its basic concepts.

M.M. LEHMAN
Imperial College of Science and Technology
London, United Kingdom

Parallel Processing: The Transputer and Occam. By A. Carling. Wiley, Bognor Regis, West Sussex, United Kingdom, 1988, Price £12.95, ISBN 0-850580774.

This slim volume is advertised, on the back cover, as providing a clear and detailed introduction to the concepts of parallel processing. The book has two distinct parts: an introduction to parallel processing (pages 1–100), and a description of the

Transputer and occam (pages 101–147). These two parts are not well integrated, and differ considerably in their approach.

The introduction to parallel processing is fairly broad, discussing applications, architecture, and software issues. However it is rather uneven in level of detail, and somewhat disjointed. There is a tendency for the text to degenerate into a list of terms which are then defined, and the ordering of the material is not always natural. It is quite common to be told the same thing two or three times, in almost identical words, and this occasionally involves ideas which are not properly explained. For example, the penultimate paragraph of page 58 starts "To enable near zero wait states...", and then talks about the use of a cache. The top paragraph on page 45 similarly starts "To achieve near zero waits...." Cache stores were explained earlier but with no mention there, or anywhere else, of what is meant by "zero wait states". There are occasional surprises; for example occam is quoted as an example data flow language! In describing various architectures, there is a move from the historic, namely a fairly detailed description of STARAN followed by one of Illiac IV, to the futuristic, in the form of a discussion of the advantages of dynamic reconfiguration, without sufficient explanation of the gap in time and implementation status. To my mind these defects make this part of the book unsuitable for academic purposes.

The description of the Transputer is very rosy, and reads rather like sales literature for Inmos and other companies, such as Meiko and FPS, with Transputer-based products. Unfortunately, like much sales literature, it is rather uncritical and also liable to becoming rapidly out-of-date. Thus recent Transputer products (such as the T800) are not mentioned, while fairly detailed information (including such things as power dissipation) is provided about some earlier ones, namely the T414 and T212 Transputers. Here also occasional errors undermine confidence, such as on page 120 "... the high performance rate of 3 byte/sec. (is thus) able to be attained." The chapter on occam inevitably reads rather like a programming manual, and suffers somewhat from the fact that the language described is the original occam, rather than occam 2, the revised version of the language. There are also a few careless errors, such as inconsistent indentation in some occam programs and use of ":=" in equality tests (on page 141), which can mislead the innocent.

A major problem in reviewing this book is in identifying the intended readership. It may be the sort of book one would recommend to a busy but nontechnical manager who wants exposure to some of the issues. Then the uneven style might be appropriate. But I would suggest that most readers of this journal could find better books on parallel processing and more up-to-date sources of information about the Transputer and occam.

C.C. KIRKHAM
Department of Computer Science
University of Manchester
Manchester, United Kingdom