the energy landscape as if they were a problem for Hopfield nets, although it is only because of their existence that patterns are retrievable. So-called *spurious minima* are the problem and probabilistic networks help to shake the net from these (relatively shallow) minima, to deeper minima, not to a global minimum as the author implies.

Since the text is for the technically-minded general reader it presumes no particular mathematical knowledge. However, little attention is paid to the difficulties which the absence of such knowledge might present. In his earliest exposures to mathematical derivations in the text, the general reader is left without even the odd 'where' or 'therefore' to light the way. Unlike many such texts, no mathematical appendix is offered for reference.

The book has the potential for confusing and misleading, and is not one which I would recommend as a starting point.

T. W. ROUTEN
De Montfort University

ROGER S. PRESSMAN (adapted by DARRELL INCE)
Software Engineering: A Practitioner's Approach. McGraw-Hill. 1994. ISBN 0-07-707936-1. £22.95. 801 pp. softbound.

This is the European, and scarcely changed, version of the third edition of Pressman's *Software Engineering*. Current trends in this subject point to textbooks weighing several tons by the next millenium, but the increment from the (1992) international third edition to this one is tiny. Darrel Ince has added a few words on software quality, including ISO 9001 (a topic as fascinating as watching paint dry) and has rewritten the section on formal methods, mostly Z (wisely taken from Spivey's 1990 paper in *IEEE Software* for the international edition). 'European' does not mean 'British English'.

Pressman's original text was born in 1982, and has thriven and spread ever since. But, as the weight of these general texts increases, their content thins. If you wanted to know about any of the topics covered—say, database normalization, data structures, object-orientation, or CASE—you would not read a book on 'Software Engineering'. And if you wanted a general overview of Software Engineering, you would not read 800 pages. However, if you really do want one of these massive tomes, Pressman's is as good as any and better than some.

But what has Ince done for the book? Not enough to justify a new edition; and worse: once again we have claims for rigour made by a writer who has not exercised even *care*. The term 'subset' is introduced without definition, and the *subset* and *proper subset* operators are given the interpretations of each other. The *union* operator is carelessly called 'intersection' and defined thus: *The ... operator takes two sets and forms a set that contains all the elements in the set with duplicates eliminated.* The conditional ('if ... then') operator is (as ever) misleadingly called 'implies' and is introduced without definition. On sequences, we read: *Since a sequence is set of pairs, then all the set operators described in the previous subsection are applicable.* If you can ignore the syntax of that sentence, and think about its meaning, you will appreciate that 'applicable' is used in a Pickwickian sense: without closure. A sequence of n things is defined as a set of n ordered pairs, the first members of those pairs being the integers 1 to n. So you can form (say) the union of two sequences, but it is unlikely itself to be a sequence. As it happens, Ince's carelessness provides him with a logically impeccable defence. The subsection on sets is not 'the previous subsection', which is about logic, and describes no set operator.

In the international edition, Pressman wisely says that *[a]mbiguity, incompleteness, and inconsistency can be discovered and corrected more easily* using formal methods. Ince substitutes the claim that *there is little possibility of ambiguity, specifications can be mathematically validated for contradictions and incompleteness, and vagueness disappears completely.* If you are waiting for Ince's proof procedure for set theory, don't hold your breath. Indeed, why not fill in the time by working as a much needed technical proof-reader for McGraw-Hill?

ADRIAN LARNER
De Montfort University

GUSS J. RAMACKERS
Integrated Object Modelling. Thesis Publishers Amsterdam. 1994. ISBN 90-5170-244-2, 255 pp. softbound.

In this book, the author has provided a clear exposition of an executable (and therefore validatable) formalization of IS/IT business analysis systems. A large proportion of the book attempts to prove that the proposed modelling formalism is beyond the reach of most case tools and thus requires an extension of such technologies. The major benefit of the proposed approach over existing case technologies is the facility to allow several different views of a system corresponding to the various users of such systems.

The described modelling framework combines all the familiar business and structured/MIS diagrammatic notations, as well as textual descriptions (as alternative representations). Underlying this formalism is a mathematical structure known as a high level Petri Net. This is essentially an algebraic extension of basic Petri Nets allowing for the denotation of objects *a la* O-O analysis/designs. Thus in addition to validation, i.e. Petri Nets are executable, due to the mathematical semantics of Nets, verification of internal consistency is also facilitated.

As a modelling paradigm there is no doubt that the present approach makes a significant advancement in