are especially looking for metrics that can provide early feedback on future maintenance and implementation costs. This is one of their key criteria for metric validity which has to be demonstrated empirically.

The book starts with a good historical review and then looks in detail at the validation of three very well-known metrics: Halstead, McCabe, and Henry and Kafura's information flow measure. This selection is rather strange given the book's emphasis on early life-cycle metrics, because the Halstead and McCabe metrics are only available after coding. There has been extensive recent activity by the likes of Kitchenham and Kemerer dealing with validation of function points (which are available at the specification stage). Since, function points are also far more extensively used than the other metrics, it is a shame that the authors did not look in detail at their validation. The authors conclude that the three chosen metrics are 'bad': not by bad luck but by carelessness of approach and poor awareness of measurement. To address these problems the authors suggest three issues that need to be addressed:

1. The meaning of metrics (via measurement theory).
2. Model evaluation (both empirical and theoretical via axioms).
3. The need to tailor metrics for different goals.

Thus the book contains chapters which deal with each of these issues in turn. Based on these rigorous principles the authors propose their own formal models of designs and define measures which can be extracted from these. The two type of validation (via axioms and empirically) are performed and the results of these lead to model refinement and ultimately to increasingly promising results.

This is a well written book which will appeal primarily to people who already have some experience of software metrics. Because of the central role of the notion of formal models and axioms, readers will need to have some mathematical maturity if they are to really benefit from it.

NORMAN FENTON
Centre for Software Reliability, City University

JIM INGLIS

C is a programming language so low-level that its identifiers designate storage, which is (they tell us) the hardware component used to implement programming variables. There is nothing wrong with low-level languages (especially portable ones), in their place; but C is also a language so execrable that any adequate specification of it is quite sufficient to condemn it, for those that have ears to hear. Can there be any argument for a professional corporate programmer (a commercial, and probably COBOL, programmer) learning or using it? Yes; and for what it is worth, here it is.

Professional programmers are not 'professionals'; they are, and ought to be, like professional carpenters, artisans that exercise their skills for money. A carpenter might properly convert from plywood work to chipboard work: you want tacky? you want to pay for tacky? OK, mate it's no skin off my nose. Is there money to be made, or a job to be kept, by writing C instead of, or as well as, our current languages? Fine, let's learn C.

If you swallow that, and you want to convert to C from COBOL (or a similar language), you could buy a copy of this book.

But if you love plain English, well written; or elegantly used typography; or limpid, systematic exposition; or shrewd and balanced judgment: you should buy two copies and give one to a like-minded friend.

ADRIAN LARNER
de Montfort University

ANDREW DILLON

This book is about reading. Andrew Dillon is an applied psychologist and a researcher in the Human Computer Interface (HCI, although he remarks, 'this acronym was often shortened to CHI'-think about it). 'Reading' (not in the index) is used in a sense somewhat wider than we might expect and includes navigation around a text either on paper or held electronically, principally in hypertext.

Dillon refers to 16 papers of which he was the first listed author and others of which he was an author. These are among his more than 200 references; but then, his researches show that 73% of 'readers' of academic journal articles access them as 'background material for work purposes' (know what I mean?), but only 46% read them to keep their knowledge up to date. Perhaps he thought the time had come to summarize all his research in a book, padded—as is all too common—by a massive work: you want tacky? you want to pay for tacky? OK, might properly convert from plywood work to chipboard work.

But if you love plain English, well written; or elegantly used typography; or limpid, systematic exposition; or shrewd and balanced judgment: you should buy two copies and give one to a like-minded friend.