width font would have been preferable). The index is very short and omits many topics I wished to cross-check.

The full code for the 'assembler' program is printed, but no disk containing the source is provided, nor is any FTP site named from which it might be downloaded. There is far too much material to type in yourself, and so testing—and experimenting with—that program is not feasible.

Finally—and perhaps most serious—some basic techniques for 'good programming practice' are ignored. For instance, `malloc()` returns a pointer, or `NULL` if it fails to allocate the requested memory. The code fragments exhibited which use `malloc()` never test the return value to ensure that it is not `NULL`.

This book is very competitively priced, and notwithstanding my comments above—which would not greatly affect a novice programmer—I consider it a very good introduction to the C language, and excellent value.

MIKE JOY
Warwick University

JOC SANDERS & EUGENE CURRAN
Software Quality. Addison-Wesley. 1994. ISBN 0 201 63198 9 £19.95. 179pp. hardbound.

The readers of this journal must include many that, like your reviewer, have devoted most of their working life to the writing of high quality software. And yet, presented with any book titled 'software quality' our hearts sink.

This book does not disappoint our expectations. It comprises a manger's guide to implementing a quality system, a software engineer's guide to 'best practice' (in 'quality', not best practice in software development), and six appendices, including overviews of the ISO 9000 standards and the Capability Maturity Model, ISO/IEC 9126 definitions of abstract nouns that mostly end in 'ility' and a summary of no less than 141 essential practices.

The managers are given the usual motivational stuff. what we used to call '*the cost of quality*' is now called '*the low cost of quality*', although it is the cost of doing things later classified as '*essential*'. We are reminded that ISO 9000 certified companies demand similar certification of their suppliers, so you better get certified and make similar demands on *your* suppliers. '*Empowering employees*' is there but only in the non-threatening sense of delegation of '*authority and decision-making ... to the appropriate level*'. '*Quality must be accepted by everyone*'; but not to worry: this can be achieved by a '*lively and appealing education programme*'. Then you can implement your quality system, get your ISO 9000 certificate and be in on the racket. Quality means points; and you know what points mean: the Baldridge Award, the Deming Award, the European Quality Award.

For the software engineers, or rather, their project managers, there are some really sensible remarks: software need to be designed for change; support is needed as well as development; the causes of problems need to be identified and eliminated. And then we have three chapters that are, in substance, a process checklist, each item being boldly labelled **essential**, **important** or **useful**; the rationale for the classification is opaque. The checklist is written, largely, as continuous prose. Here is a specimen: '*It is important to identify in the acceptance test plans the acceptance tests necessary for the provisional acceptance of the software*'. Get the flavour?

'*It is clear*', say the authors (not descending to argument), '*that the quality of software is largely determined by the quality of the process used to develop ... it.*' It is indeed clear, because it is trivial, if by '*process*' one means 'policies, procedures, tools and resources, both human and technological.' But if one means process—the sort of procedures that have to be recorded for, say, ISO 9000 certification—then it is not clear at all. Much quality depends not on process but on the professional exercise of skill and care: do you choose your hairdresser, dentist, car mechanic or dentist for their process quality? And is it, do you suspect, just possible that—even in these latter days—the most important thing to do if you want high quality software is to employ some really clever professional software writers?

But the authors have anticipated your reviewer: '*It is common for software developers to resist written standards and procedures...*' Then they reject them out of hand. That is '*software quality*' for you: those that understand high quality software, and have long been committed to it, are to be educated in a lively and appealing way, given a fair hearing, and *humoured.*

ADRIAN LARNER
De Montfort University

S. LAKSHMIVARAHAN & SUDARSHAN K. DHALL
Parallel Computing Using the Prefix Problem. Oxford University Press. 1994. ISBN 0 19 508849 2 £45.00. 294pp. hardbound.

This book comprehensively surveys parallel prefix algorithms for shared memory and circuit models. It is probably the only book available in the literature for such a detailed treatment of the subject. The book begins with the prefix problem and its applications and gives an overview of the parallel models (shared memory and circuit model). The subject matter for these topics is very well written and contains several good examples, many somewhat theoretical.

The second set of chapters covers algorithms for shared memory models, more specifically, parallel prefix algorithms on arrays and on linked lists, which include Schwarz's algorithm, Cole-Vishkin algorithms as well as randomized algorithms. Several examples are given for the reader.

The rest of the book focuses on parallel prefix algorithms for circuit models and discussion of fan-in and fan-out restrictions for the circuits. The analysis of

the circuit algorithms (e.g. Ladner-Fischer circuits) is excellent. There is a detailed discussion on size versus depth trade off in the circuits as well. Finally, the appendices contain useful material on semi-groups and monoids as well as other material relevant to the book.

The book is quite theoretical but is self-contained, thanks to the appendices. To apply the algorithms to message passing architectures will need some thought on the reader's part. While the book has an extensive coverage of algorithms and their analysis as well as several examples to explain the algorithms, I feel that more intuition is needed as to why the algorithms work. For example, rather than providing the algorithmic complexity and an example table it would be helpful if the authors highlighted the fundamental reasons why the algorithm works. The treatment also needs to be more cohesive (in particles for circuit models) to show how the algorithms relate to each other. Otherwise the reader might get the feeling of reading a disjoint set of algorithms and not be able to gather the fundamental commonality or differences among them. Further, any background material covered in the text should state upfront where it is going to be applied.

For these reasons I feel that the book is more appropriate for research purposes or an introductory graduate level course; it may be difficult to understand at an undergraduate level. At the research level I think it is a very good book to have on one's shelf if one needs to apply parallel prefix computations. The choice of algorithms as well as their analysis is excellent. For the research student there are a good number of exercises at the end of each chapter; the algorithms are also presented in a form that can be easily implemented. The notation, too, is very readable.

RAJESH K. MANSHARAMANI
Tata Research Development and Design Centre
Pune, India

GERARD TEL
Introduction to Distributed Algorithms Cambridge University Press. 1994. ISBN 0 521 47069 2 £29.95. 534pp. hardbound.

Distributed systems form a wide area and to choose what should go in a book and what should stay out is hard, especially since no particular 'core' has been defined or agreed upon. For example, though networking is now generally accepted as distinct from distributed systems, there is still some confusion about exactly how much of networking should be considered as part of distributed systems. A clearer identification of the boundaries of distributed systems would be very useful for the discipline. This book does not help much in this regard. In fact, a fair portion of the book is devoted to protocols that are used in networking and covered in books on networking (though typically with lesser 'formality').

A book on distributed algorithms can focus on algorithms of a particular type, or select some areas and cover algorithms in that areas. This book has taken the latter approach. The book is organized in four parts, plus an introductory chapter which is basically a collection of brief overviews of many different topics. Unfortunately, it does not help the reader much in identifying the scope of the book.

Part One deals with algorithms that are used in communication protocols at different levels. The sliding window protocol and algorithms for routing and packet switching are described. However, protocols for reliable or atomic broadcast, which fall more under distributed systems than networking, have not been discussed.

Part Two is closest to what might be considered the 'core' of distributed algorithms. This part includes chapters on wave and traversal algorithms, election algorithms, termination detection, snapshots, etc. (though algorithms for mutual exclusion, an important and a classical problem in distributed systems, have not been discussed). Each chapter describes various algorithms that have been proposed.

Part Three focuses on fault tolerance. Fault tolerance in distributed systems is a wide area (there is a book devoted to just this topic [1]), so clearly cannot be covered in breadth in a book of this type. This part of the book, in fact, focuses mostly on Byzantine agreement, clock synchronization, and self-stabilization, and leaves out the topics of state restoration, commitment, atomicity, replication management, etc. Part Four contains appendices.

Overall the book is a reasonable, though not complete, reference for various distributed algorithms. There is a fair amount of formalism (a bit too much for my liking).

[1] P. Jalote, *Fault Tolerance in Distributed Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1994.

P. JALOTE
Indian Institute of Technology,
Kanpur

SIEGFRIED TREU
User Interface Design & User Interface Evaluation. Plenum Press. 1994. ISBN 0 306 44681 2 & 0 306 44746 0. $79.50 & $69.50. 351pp & 282pp. hardbound.

Treu's two-volume set is intended for use as either a text book, or a reference tool. The first thing that a potential reader should be aware of is that whatever is implied by the title, Treu himself states that this is not an implementation document—if you are looking for something as a guide on good implementation style stop right here. There is no advice on screen layout, aesthetic issues or cognition in these books and as such they are directed more towards the task of system design than implementation of the actual interface. This really contributes to the feeling that they are possibly a little incomplete on the subject at hand.