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.

The first book starts with an extensive background on the history of HCI and user interface design. It is questionable whether this is really warranted, especially at the level of detail at which it is presented. Most readers will not be too interested with the various name and acronym changes of the HCI special interest groups in '*the early days*'. That the author met Doug Englebart (best known as inventor of the mouse) and found him a nice unpretentious bloke is an example of a questionable inclusion. Another thing that is noticeable at this stage is Treu's passion for formal definition and acronyms. It seems that in this book, if there is an acronym for something then it is defined and used as such, even if it occurs only once. Additionally, every single operative word is formally defined—even when the definition is identical in meaning to that which any pocket dictionary would give. Opening the book completely at random as I write gives me a definition of the word 'context'. I know of no one that does not understand this word in the sense it is given: is this padding or does the author really feel a need to make sure his readers know what this means? Trying to get into this book is not easy, and this is entirely down to the fact that it reads like a set of old lecture notes. Flow and continuity is poor and at some stages you really feel like putting it down, even in the middle of the more interesting portions.

If you persevere, however, you will not be entirely disappointed. The books contain some interesting subjects. One thing that a reader will notice is that they are well researched. There are further references everywhere (although at some times this proves to be a further obstruction to smooth flow) and the bibliography in the first book alone runs to some 12 pages. The second volume gives the number of publications selected and referenced for the two book set as 'over 425', and goes on to briefly discuss trends in those works. This would supposedly cover the bulk of significant published material on the subject in its short history. As such, anyone looking for a starting point for research or directed reading in this area could do far worst than look at Treu.

If you survive the early chapters and acquire sufficient resistance to the style issue, you go on to find that the books do indeed offer some worthwhile insights into the field. As I have said, Treu does not concern himself with issues of layout, but more at the feature level of interaction. How to analyse what constitutes a necessary inclusion in an interface design and what will produce a good interface between software and people is the name of the game, even if it is in a somewhat formal style. You might go as far as to describe these, indeed most HCI texts, as attempts to make order from chaos—taking the highly subjective issue of interface design and giving it a more formal, logical, structure. Do they succeed? Well, this itself is a subjective question, but I think that they could certainly have done a worse job.

As a reference tool, the books are good, but more as a guide on where to look than a definitive source of answers. How does it stand as a textbook? I would say 'on a somewhat less stable footing'. It is not easy to read Treu, and certainly many of those at the undergraduate level will struggle to find the willpower to read the books from cover to cover, or indeed to finish them at all. The numerous exercises in the book are best avoided as presented. As something to provoke thought on each chapter they are fine at the mental level. I pity anyone called upon to answer them textually though—potentially a very tedious process, and one that could not be accomplished without repeated back-references. If used as such, the only purpose served by the exercises is to make the reader *read* the book, only benefiting those who would otherwise not so do. I would say that the books could be used for a very specialized course—'only those *very* seriously interested need attent'. Additionally, a complete HCI course would need a sizeable amount of supplementary material to cover the absence of aesthetic and cognitive-based considerations on the more 'human' side of the subject.

Who will want to read this book? Basically, anyone looking for a place to start, or interested in all sides of the subject. Those strictly into implementation and programming are advised that they may get ideas from this book, but not direct help, and if you want to know how to explicitly lay out a screen, try looking elsewhere. Mr Treu does, after all, advocate going out and asking the advice of some 'experts' in the relevant field as a vital step in designing an interface.

COLIN ISAAC
Warwick University

KEE YONG LIM & JOHN LONG
The MUSE Method for Usability Engineering. Cambridge University Press. 1994. ISBN 0 521 47494 9 £29.95 330pp. hardbound.

MUSE is an acronym for Method for USability Engineering and this book is about the design process in human–computer interaction. Lim and Long argue very powerfully for a human factor design cycle which is complementary to existing methods of structured design for the functional aspects of a computer system. Unlike some sets of HCI design guidelines the MUSE methodology attempts to express design in procedural rather than in declarative terms and this approach lends itself to the integration with structured software design methods.

The contribution of human factors to system development is characterized as '*too little, to late*'. The book focuses on the development of a method for integrating HCI into the early stages of the system development life cycle, where errors are the most expensive to correct. This is in contrast to some other techniques which tend to raise HCI issues only towards the end of the life cycle. In other words, MUSE has great