of his close involvement with the standard for C, are both entertaining and instructive—and should be required reading for all who argue about language standards.

If I have any complaint about this book, it is just that the price seems a little high. Maybe the intended audience has become accustomed to paying this sort of price for more technical books and the publisher does not expect large sales. Perhaps the strongest recommendation I can make, therefore, is to say that I went out and bought volumes I and III on the strength of this volume, and I look forward to further hours of educational entertainment.

C. C. KIRKHAM
Manchester University

MARK NORRIS, PETER RIGBY AND MALCOLM PAYNE
The Healthy Software Project. John Wiley & Sons, 1993, £24.95, 198 pp hardbound, ISBN 0-471-94042-9

The authors of *The Healthy Software Project* are employees of British Telecom and have extensive experience of project management using process quality standards, principally ISO9001. They present some valuable learning points, gleaned from their experience and scholarship. Readers that have worked in software development may suffer, as your reviewer did, the aching of old scars.

The book, as its title indicates, is structured around a medical analogy: the authors have, mercifully, resisted the temptation to call it a 'paradigm'. Its eight chapters introduce projects and the analogy; discuss the special characteristics of software projects; describe some projects (good, bad and average); and cover symptoms of project 'disease', therapy (including euthanasia), recovery and health maintenance. Three appendices treat of quality management systems, product metrics, and a process applicable to project recovery. The analogy is strained at times, even to the inadvertent black humour of: 'As with people, it is the quality of life ... that is important not mere survival. An invalid software project can cost a fortune to keep.'

However, the most important message of the book—often ignored in both theory and practice—is that projects can be, and sometimes should be, *recovered*. Project management is not merely about adequate planning and monitoring; it must also encompass recognition of when things go wrong (they usually do), and what to do to put them right. Starting with a clean sheet is as rare a privilege for a project manager as it is for a software designer.

The authors' attempt to present their learning points as an ordered collection is not entirely successful. We find a number of radical remarks that are made once and then ignored: 'A QMS ... provides a basic level of project control, but no more'. Yet the 'therapies' proposed all seem to be process ('management') therapies: for 'sick' projects, we need 'the necessary control mechanisms to recover them'. Most radical (and very true): 'the production of software is a creative process and is usually done on the roll' (i.e. in a sustained, enthusiastic burst by skilled and committed software writers). Quite: the authors give lip service to 'people' (along with process and product): but in none of their case histories do they mention any of those extraordinary, talented, and often wild people that write the best software, and recover failing projects, technically. This is unquestionably a book by and for project managers: it won't fool your techies any more than you do.

On metrics, the authors quote Professor Dijkstra ('Metrics is crap.') but decline to engage with his arguments. However, they remark that 'many product measures ... are currently offered by the Snake Oil Salesmen ... the consultants and the tool vendors, with very little evidence that what can be measured is ... useful to measure'. Would that we could make the right judgement about metrics merely on the basis of who promotes them: depend on those recommended by employees of large corporations who get no commission on sales?

Finally, a plea to the publisher: proof reading is as important in your trade as testing is in ours. The reader should be spared, at least, poor spelling ('euthenasia', 'committment') and punctuation ('the customers aspirations'); although the correct placement of 'only' is nowadays perhaps more than we could hope ('that will only be achieved through projects ...'). Even more constructive editing might have clarified, for instance, the ambiguous 'key drivers': does 'key' mean 'major', 'important' or 'essential'? does 'drivers' mean 'resources', 'characteristics', or merely 'factors'? Perhaps the authors could advise on TAM (Total Quality Management—neither in the glossary; indeed—and for shame—there are no acronyms in the glossary, and not all are in the index).

A. LARNER
De Montfort University

BONNIE A. NARDI
A Small Matter of Programming. MIT Press, 1993, 162 pp, hardbound, £29.95 ISBN 0-262-14053-5

In 1979, Bonnie Nardi tells us, she spent several months in Papua New Guinea and came back with an account of how its villagers communicated with slit-gongs. This book is her account of what she has learned in several years of studying the communications of another exotic folk: users of application software packages. The burden of Nardi's account concerns the amazing ability of these people to master formal systems and notations, and the ways in which they co-operate with each other in their spreadsheeting and computer assisted design.

These topics are—or ought to be—of vital interest to the designers of software packages. However, plenty of advice on usability has already been proffered to the

designer (and, judging by her extensive list of references, read by Nardi). Is there any reason why we should welcome yet one more (blessedly slim) volume?

There is every reason. Nardi is a natural historian of humanity, with acute powers of observation and analysis, and—which is even rarer—the ability to marshal evidence and conduct argument. In her discussions of the mundane human conversation as a model of the human/computer interface, and of a variety of interaction techniques, she wields the blade of reason so courteously and delicately that even those whose ideas have been cut to size, or given an elegant *coup de grace*, could scarcely complain at the treatment. But between these discussions comes Nardi's major achievement: a beautifully sustained and forceful argument that ordinary human beings can master formal systems, and even special notations. Now recall all those other approaches to usability, including not a few among the writings of the Artificial Intelligentsia, that start from a presupposition (rarely evidence or argument) to the contrary.

We are offered here a collection of well founded and immediately applicable advice by a skilled advocate of the user. While most of that usability literature may safely be left on the shelf, the package designer that does not now read Nardi will be without excuse.

A. LARNER
De Montfort University

DOUGLAS E. COMER AND DAVID L. STEVENS
Internetworking with TCP/IP Volume III: Client-server Programming and Applications. Prentice Hall International, 1994, £22.95, 508 pp softbound, ISBN 0-13-097627-X

Volume III of this now established series on the Internet protocols augments Volume I (which explains what the TCP/IP network is) and Volume II (which shows how it works) by giving an extensive and readable account of how applications may be built to use TCP/IP.

As TCP/IP was loosely specified to allow it to run on a variety of operating systems, there can be (and are) different application software interfaces with TCP/IP protocol software. However as the authors state, in practice there are only a few. The two main interfaces being Transport Layer Interface (TLI) and the socket interface, companion versions for Volume III were written, one for each interface. Much of the material is of course common to both versions, but as any one site is likely to use only one interface, having different versions has the advantage of hiding irrelevant details of the interface one is not using.

Internetworking with TCP/IP has several introductory chapters, which give a general overview of TCP/IP and client-server programming in general. More specialized chapters follow on NFS, telnet, etc.

I found the descriptions offered very clear and unambiguous. Important concepts are emphasized with short summaries of a few lines in italics. Algorithms are numbered and placed in boxes for easy identification where they are referenced. In the later chapters example code is presented, which is always useful. I received the TLI version, which had a handy appendix describing in detail all the TLI library functions, I presume the sockets version has a similar section.

The structure of the book has clearly been given thought. Cross referencing between chapters is very adequate, the contents, bibliography and index are excellent, and chapters are short enough to be read in one sitting. Each chapter starts with a brief description of what chapters prior to it are directly relevant, and which follow on from it, which for readers such as myself who browse a lot is handy, and exercises are offered on each chapter.

I particularly liked the simple map at the beginning of the volume with internet and electronic mail connectivity shown. While it is the nature of the internet that this will (hopefully) be out of date almost immediately, it does show yet another east–west north–south divide. For almost all eastern Europe and much of South America has no internet (but do have e-mail) and Africa alone is singled out as almost entirely lacking in internet connectivity (except South Africa) or even in most cases e-mail connectivity.

I am not expert in networking (though colleagues of mine to whom I have shown this volume are and have found no major errors), but I found the text easy to understand as it is a coherent, cogent and lucid presentation of what is often seen as a rather arcane subject. At £22.95 for a fairly extensive treatment of the subject (508 pages) this volume is inexpensive as technical texts go. The former two volumes have become the *de facto* standard references to TCP/IP, and there is no reason to believe this volume will not fulfil the same role for builders of applications using TCP/IP.

D. ANTHONY
Warwick University

CHRISTOPHER BABER AND JANET M. NOYES
Interactive Speech Technology. Taylor and Francis, 1993, £37.50, 212 pp, hardbound, ISBN 0-7484-0127-X

This is a useful collection of articles about the problems involved in using existing speech technology (speech synthesizers and speech recognizers) in realistic applications. It does not have much to say about how speech synthesis and recognition works, but concentrates on speech-driven HCI. Many of the contributions were first presented at a one day conference at the NEC Birmingham. Most of the major UK laboratories involved in this kind of work are represented, but there is no international dimension apart from the foreword. The editors have put considerable effort into a series of linking articles which introduce groups of individual contributions.