

ment of requirements and so on. The list of elements could of course be adapted to the special requirements of a given project.

Although I was disappointed that the book was limited to these two costing techniques, Wellman makes up for this by presenting a considerable amount of practical advice about project management, especially in areas related to cost control. The style is quite down-to-earth. The chapter on tendering should be required reading for clients who do not really know what they want until the contractor tells them what they are going to get.

MIKE FIELD

THE RAISE LANGUAGE GROUP

RAISE Specification Language. Prentice Hall. 0-13-752833-7. £24.95

I like this book. It is about a formal method which draws on the strengths of the established methods (like VDM) and avoids some of their weaknesses. The book presents its subject matter in a straightforward, practical way. It does not try to be obscure nor to impress the reader with the cleverness of the ideas it describes. However, I have my doubts about its place in the BCS Practitioner Series: these doubts are as much because of my assumptions about the background of BCS 'practitioners', and how much time they might have to read a book like this, as because of the approach of this book.

Let me clarify the assumptions before discussing the book. When I see the series title BCS Practitioner Series, I think of busy software professionals and their managers. They are under pressure, dealing with new hardware and software technologies and with innovative developments. They are constantly under pressure but are acutely aware of their need to keep up with their subject—and the subject formal methods is one which frequently bothers them; new graduates keep mentioning them in their job applications and documents from government agencies frequently assume the need for them. So, when browsing the formal methods books, the RAISE Specification Language looks interesting: it does not have 'formal' in the title, when rifling through the pages, it appears to have fewer mathematical symbols than neighbouring VDM or Z books, and looking at the first chapter, the specifications look like programs and are presented in a pacey style.

For me, these apparent pluses turn out to be minuses—not big minuses, but minuses nonetheless. The book promises a tutorial part, a reference part, and extensive appendices. Here is the rub. The 'tutorial' is a fast run-through of most parts of the RAISE language—basic concepts, built-in types, products, bindings, functions, sets, etc. Unfortunately, for the 'practitioner' it does not take time to carefully relate these concepts to practical (traditional) software development. It is left to the busy practitioner to work out how these specification

techniques can be used. In fairness to the authors, that is all that is promised on the back cover of the book. But more is needed, and more is expected when the term 'tutorial' is used in the first chapter where the structure of the book is described.

As a reference book for the RAISE specification language (RSL) the book is excellent—this is expected as the authors are the RAISE language group. The 'tutorial' gives syntax and examples of all structures. Initially these are given top-down; later more complex constructs for concurrency and information hiding are described. The 'reference' part deals with expressions, naming and binding in depth. Finally, quick reference syntax and lexical definitions are given.

To sum up: if you are a practising software engineer, or a manager of software engineers, who has not yet encountered decent explanations of how a mathematical approach to specification and design relates to traditional software development, you would be well-advised to read an easy-going introduction to formal methods (which probably uses VDM or Z) before reading this book. If you do that, you will certainly benefit from a clear practical reference book for the Raise Specification Language. On the other hand, if you already 'into' formal methods, this is an excellent source for RAISE.

M. WOODMAN

RUSSEL WINDER

Developing C++ + Software, 2nd edn. John Wiley. 0-471-83610-3. £19.95.

C++ is becoming the language of choice for a large group of problem solvers. It is not surprising that C++ texts abound in the market. Winder provides us with a good one, and reasonably priced, as well. It reflects the richness of C++, and teaches *that* language, not a jumped up C. One small example: the **struct** construction, while it exists in C++, is not mentioned, as **class** handles the same notion.

This is a text written in the first person, not a reference. To be precise, it reads as if it were a set of lecture notes (as indeed Winder states). This has its disadvantages. It is a *very* linear book. Even with the use of the index, it is hard to find expositions of a particular language aspect without a lot of reading. Winder is forced to use concepts without prior explanation. As an example, **enum** is used on page 123, in classes, but not properly introduced until page 140. Sometimes such an ordering is unavoidable, but **enum** is a useful scalar concept in its own right. Another is the quick introduction of ADT's on page 9, which are used for the first time in classes on page 123. On the other hand, the notion of **friend** is introduced in section 8.4 with classes, where it confuses. It is not needed until inheritance, where Winder deals with it well.

The book displays the context in which it was written, this being a course for conversion MSc students and for

first year students whose first language was Miranda. Some of the reasoning necessary for good programming might have come from the Miranda course, which is all to the good. However, perhaps that is the reason for the sudden and unexplained appearance of \forall on page 96. Given the right context, I would have liked more such specification. Unfortunately, one cannot assume that context.

Each chapter is introduced well. It would help to have a summary of the notions he deals with at the end of each chapter, particularly in so very linear a book.

Two major developmental notions are introduced and then dropped. I agree that Design Structure Diagrams can be a useful tool. Winder's treatment in the chapter on Sequence, Decision Making and Iteration is very lucid. The technique should have been used in later chapters as well. He introduces complexity at the end of that chapter, and again, does not use it in the rest of the book.

There are some minor points. On page 104, Winder uses the fact that the `&&` operator, unlike most operators, defines its order of evaluation; a note that this is dangerous is buried on the bottom of page 105. The functions on page 129 should return `void`. Chapter 11, Dynamic Data Types, is too long (as Winder states). Perhaps he could insert a separate chapter with the extended example.

The book has a very good set of exercises. It could do with a few more. They are scattered through the text, elucidating the idea at that point in the text. They are all answered in the back of the book. I also particularly appreciated the very good annotated bibliography. The annotations have clearly been coded. It is easy to see which of the entries Winder dislikes.

This is a good text that will appeal to serious beginners. It will help them travel well down both the roads of software development and C++ technique.

DAVID LYONS
Essex

P. K. MCBRIDE

C CLEARLY: An Introduction to C Programming. Blackwell Scientific, Oxford. 0-632-03395-9. £14.99 (Paper).

The rubric on the cover indicates that this book is suitable both for the 'absolute beginner' and for people 'experienced in other languages'. I felt that a novice programmer would quickly be lost and that a significant background in programming is essential to understand the book fully.

All aspects of C are covered in no more than 200 pages and the style is very concentrated. The text is liberally sprinkled with interesting examples, most of which are of technical interest to computer scientists, and topics such as sorting arrays and implementing stack and trees are covered. The examples are clear and

easy to follow, though an unusually small font was chosen for the segments of code and for that reason I found it tiring to read.

The book commences with a discussion of simple programs with rudimentary input and output, building on this to introduce conditional expressions and loops. Functions are then discussed, followed by complex data structures. The author then returns to the topics introduced previously and discusses them in greater detail. Pointers and strings are covered last, and this part of the text I found quite hard. However, several sizable programs are included which, although themselves moderately complex, are well set out and straightforward to follow.

In the first appendix, a list of the standard library of C predefined procedures and functions is provided. Where appropriate, a back reference is made to where they were introduced earlier in the book. In two further appendices general rules and hints are given for translating programs written in Pascal or BASIC to C.

I felt that too much material was compressed into too small a book. If you have programmed in Pascal or Basic, then this book will help you to convert to C quickly; if you have never programmed before, it is probably not a suitable first text. Nonetheless, it is reasonably priced and good value.

M. S. JOY
Warwick

DON LIBES

Obfuscated C and Other Mysteries. Wiley, New York. 0-471-57805-3. US \$39.95 (Paper).

Over several years a competition has been held annually called the 'Obfuscated C Code Contest'. The purpose is for competitors to write programs in C whose function is as obscure as possible to someone reading the source code for the program. This has produced many programs which are extremely difficult to decipher, even for the most experienced C programmer.

This book documents the 'winning' entries for the competitions from 1984 up to 1991, and discusses each in detail. Without exception, each is fascinating and will keep the reader spellbound for hours as they first attempt to work out what it does, and then read the explanation provided. However, this book is not just for fun! The examples from the competitions are used to illustrate and to introduce many serious C programming techniques, such as using pointers to functions or trapping signals.

At 400 pages the book is big and not cheap, but is worth every penny, and comes equipped with a disc (for a PC) containing the programs listed in the book. If you program in C at all, you will thoroughly enjoy this book. If you are an expert C programmer, you will be fascinated by the variety of abuses of the language. If you value the good programming techniques espoused