VDM. (The percentages given refer to the number of pages in this part of the book. The remaining 13% is devoted to Fresco.)

The first part of the book consists of three chapters. The first of these is by Kevin Lano and Mary Tobin and it looks at what should happen in the analysis phase of software development before a formal object-oriented specification is written. Chapters 2 and 3 are by the editors and Chapter 3 considers how an object-oriented specification language can fit into the software development life cycle.

Chapter 2 is, in my opinion, the best in the book. In it the editors provide the reader with a brief overview of each of the languages used in the case studies—as well as Abrial's language Abstract Machine Notation (AMN)—but more importantly they evaluate these languages using no less than 16 well-chosen criteria. Furthermore, they identify the deficiencies of the various approaches and suggest areas that need further development. I would advise everyone working in the area of formal object-oriented specification to carefully read and study this chapter.

Because most of the book deals with languages based on Z, I think that it will be of greatest interest to the members of the Z community. One of the main problems with Z as it exists today is that officially it only allows flat specifications to be written. The language designers whose languages are represented in this book address this problem using ideas drawn from object-orientation. I believe that Z can only benefit from being enriched in this way.

The main omission from the book is a chapter devoted to Abrial's AMN and its associated B-method. AMN and B are discussed at length in Chapters 2 and 3, but no large case study written in AMN is provided. The book would have been even better than it is if the chapter on Fresco had been replaced with one on AMN.

ANTONI DILLER
Birmingham University

B. SANDEN
Software Systems Construction with Examples in Ada.
Prentice Hall International Editions, 1994, £22.95, 443 pp., softbound, ISBN 0 13 288580 8

There are many textbooks about Ada. Despite the insistence of the Ada community that learning to program in Ada is more than learning the syntax of the language, by far the majority of the books, whether they are aimed at experienced programmers or at novices, concentrate on the syntax of the language. This book joins the small but distinguished group of texts that seriously address the design and implementation of software, using Ada as the language of expression. The emphasis is on what the author calls *reactive* software, more commonly known as hard real time software.

The eclecticism of the author's approach, one the book's strengths, is made apparent in the introductory chapter, which sets the material covered in the book in the context of modern software engineering thinking.

The second chapter covers control structuring and is heavily influenced by Jackson's structured programming. An important distinction is introduced, between what the author calls *explicit mode representation*, where the current mode of execution is remembered by explicit mode variables, and *implicit mode representation*, where it is represented by the location of the flow of control in the program text.

Modularization is dealt with in Chapter 3 and the influence of Ada is, as might be expected, very evident here. Ada provides probably the best modularization facilities of any commercially supported language but much of the discussion is valuable in a wider context; in particular, the section on the use of independent subprograms is a penetrating discussion of the ways in which independent subprograms are commonly misused and of the way in which such misuse arises. Chapter 4 continues the theme of modularization in the context of object-based software construction; this again is firmly based on Ada and includes a number of illuminating and non-trivial examples—as well as the inevitable stack.

Chapter 5 introduces finite automata formally and includes a number of excellent examples and exercises, illustrating their application to software design and construction.

Chapter 6 covers concurrent tasks and Chapter 7 the related topic of resource sharing. Chapter 8 returns to the topic of entity-life modelling, now in the context of concurrent systems; it is a synthesis of much of the previous material. The final chapter is a case study of a flexible manufacturing system.

The strengths of this book are its eclecticism; the excellent and realistic examples, well integrated with the more theoretical discussion; and the discussion of practical issues and the reasons for specific design choices. In contrast with some authors in this field, Sanden is not (usually) content to give a single, simple answer which raises more questions than it answers.

The book is well and clearly written, even if the author's Scandinavian origin is occasionally apparent, and there are few misprints. Each chapter includes a well chosen list of references.

This book is based on two courses given in a software engineering Master's programme at George Mason University in the USA. It would form an excellent basis for a second course on programming and design within a software engineering Bachelor's degree in the UK but it does not correspond closely to what is usually taught. I fear, therefore, that it will not be as successful here as it deserves. It is also a potentially very valuable book for practitioners. Unfortunately, but probably inevitably, its readership is likely to be limited to those who are familiar with Ada or with languages such as Eiffel or

Modula-2 which share some of Ada's philosophy and design goals. In other words, those who might learn most from the book are the least likely to read it.

FRANK BOTT
University of Wales, Aberystwyth

BRUCE P. LESTER
The Art of Parallel Programming. Prentice Hall International Editions, 1993, 375 pp., softbound, ISBN 0 13 074980 X

THOMAS BRÄUNL
Parallel Programming, An Introduction. Prentice Hall International Editions, 1993, 270 pp., softbound, ISBN 0 13 336827 0

It is generally accepted that the speed of computers organized according to the von Neumann model is reaching its physical limit. In order to overcome this problem, efforts have been focused on the design of *parallel* machines as well as languages to program them. Especially over the last decade the significance of parallel computing has become increasingly clear. Various architectures have been suggested and algorithms have been developed to perform efficiently on them. For this reason, most universities with curricula in computer science are beginning to offer courses in this area.

Parallel programming is not an easy subject to master. It contains subtleties and complexities and it brings together elements from many different areas. In particular, writing a parallel program involves the design of a parallel algorithm, its implementation in an appropriate parallel language, debugging and performance evaluation, and at the same time a thorough knowledge of parallel computer architectures.

Many books have been written in this field, most of which are suited as reference books or for intensive graduate courses. In addition, the majority are mostly concerned with only one of the areas involved, such as parallel algorithm design or parallel architectures.

The Art of Parallel Programming by Lester is intended for a wider range of students and provides a general introduction and overview of parallel programming. With only a few prerequisites, it can be used at both the advanced undergraduate level or at the beginning of a graduate course.

Most of the existing books on parallel programming advocate the concept of abstract programming that hides the underlying architecture by presenting programs which are independent of particular machines. However, Lester stresses the importance of considering the target architecture while writing a program and throughout the book there is a continual interplay between parallel architectures, languages, algorithms and performance evaluation. The text is clearly written, well structured and organized according to major programming techniques. The material is covered thoroughly, leaving hardly any questions open and chapters are complete with summaries, numerous examples of parallel algorithms studied in depth as well as exercises and programming projects.

The feature that makes this book unique as an introduction to parallel programming is the software accompanying it. This software consists of an interactive and user-friendly system which is written in Pascal. It allows students to write their own programs in the parallel programming language Multi-Pascal, the language used in all examples in the text. Programs can be compiled and debugged with the aid of a powerful debugger and then their execution can be simulated. It is possible to specify characteristics of the required target machine such as the number of processors, whether the machine has shared or distributed memory and its topology. In this way the performance of the program can be compared for different architectures. Multi-Pascal was developed by the author and it consists of a simple set of parallel abstractions powerful to represent parallel algorithms for both multiprocessors and multicomputers.

Thus Lester's book makes an interesting and readable introduction to parallel programming and supplements reading and understanding issues concerning parallel algorithms with the experience of actually writing programs and seeing how they perform.

Bräunl's book *Parallel Programming an Introduction* is directed to a similar range of students. It pays particular attention to system architectures and how they can influence the development of parallel programs. However, whereas Lester encourages understanding of parallel programming through experience and examples, Bräunl concentrates on presenting and discussing issues of parallel programming. A number of interesting examples are included in the text but they are not discussed into much depth.

A notable flaw of the book is the following: even though it is a successful translation from German to English, its style makes it difficult to read continuously and the coverage is somewhat patchy. Often it is the case that not enough information is given to clarify figures and program listings and occasionally the tone becomes slightly instructional. Also, at points questions are left open but usually references are given to sources for further discussion. Still there are various features to recommend this book. Most of the book is comprehensible, and very informative. Problems of synchronous and asynchronous programming and possible solutions are covered clearly and at a well-judged level of detail. In addition, discussions concerning automatic parallelisation bring interesting insights to the subject. Unlike Lester's book, it contains parts on specific programming languages with information concerning their parallel constructs. A chapter of the book is dedicated to the introduction to Petri nets. Their significance as a tool for the definition of asynchronously parallel tasks is stressed but one may criticise the fact that hardly any