smooth with plenty of body. At the other end of the spectrum but equally refreshing is Prabhaker Mateti's paper specifying a 'Pascal Program Indenter', the emphasis unusually is on an actual program, and the specification is actually a mixture of post-hoc specification and verification. Unlike the other evaluations and case studies, done to prove a point, Mateti's work reads much more like a real experiment. Something all too rare in our so-called science.

The last section is on actual specification languages, and suffers especially from the absence of papers' appendices. The papers selected all show a very diverse range of approaches. It includes papers on Burstall and Goguen's CLEAR language that seemed such a powerful approach it left me wanting to know more; a description of Goguen's OBJ, an executable specification language; a specification language distributed systems that allowed execution and run-time validation; and a prototyping language. The emphasis on executable specification languages in this section surprised me. The clarity and power of the language is inevitably compromised to make it executable, making it less useful as a specification tool. It is clear however that it is a good way of introducing programmers to the idea of specification and the ideas in specification languages, as so many of them are used to understanding something by making it jump through hoops.

Inevitably a book like this is 'good in parts', but I am grateful to it for widening my view of this subject and through its references, and through introducing me to some of the major names in this area, making it easier for me to investigate some of this broadening horizon.

Tom Parke
Imperial Software Technology
Reading, United Kingdom


Barbara Liskov's and John Guttag's book addresses a difficult problem; the methodological and disciplined, indeed the engineered production of quality software. It meets this challenge admirably, thereby setting a standard both for teachers and practitioners of software engineering principles.

The issue of primary concern in the first half of the book is a methodology for constructing substantial programs, which is based on the notion of decomposition with the aid of suitable abstractions. Three types of abstractions are identified and discussed: procedural, data and iteration. The behaviour of these abstractions is defined through specifications. The application of these principles is illustrated using LARCH as the specification language and CLU as the implementation language. A useful mapping of the methodology to other more widely used implementation languages such as Pascal is presented in Chapters 7 and 15. A multiplicity of
examples and exercises is provided throughout the text. Additionally, the appendices contain a useful collection of sample programming assignments.

The second part of the book is mainly concerned with the application of these principles to all the stages of software production, from analysis and design through to implementation and review.

Chapters 10 and 11 deal with the concepts of formal specification and verification. It is a fair criticism to say that these both significant and thorny issues are not treated in sufficient depth. For instance, although LARCH is used as the specification language, no formal semantics is provided and consequently the task of showing that an implementation satisfies the specification becomes obscure. Dealing with these issues adequately for teaching purposes will require a supplementary text.

A secondary objection is the chapter sequence of the text, which should probably be relaxed when using the volume as a course text book. An alternative ordering could be

<table>
<thead>
<tr>
<th>Introduction to CLU</th>
<th>Chapter 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Types of Abstraction</td>
<td>Chapters 3, 4, 6</td>
</tr>
<tr>
<td>Specification</td>
<td>Chapter 8</td>
</tr>
<tr>
<td>Formal Specification and Verification</td>
<td>Chapters 10, 11</td>
</tr>
<tr>
<td>Software Production Process</td>
<td>Chapters 12, 13, 5, 14, 9</td>
</tr>
<tr>
<td>Other Programming Languages</td>
<td>Chapters 7, 15</td>
</tr>
</tbody>
</table>

V. STAVRIDOU
Department of Computer Science
University of Manchester
Manchester, United Kingdom


This introductory text on the complexity analysis of algorithms is attractively styled and moderately priced. It attempts to fill the need for an undergraduate textbook at a more elementary level than that of Aho, Hopcroft & Ullman's The Design and Analysis of Algorithms and Knuth's The Art of Computer Programming volumes. The list of contents shows a very appropriate choice of material with some emphasis on numerical analysis reflecting perhaps the author's own interests. To illustrate the book's scope, I reproduce the main headings:

1. Introduction
2. Evaluation of Polynomials
3. Iterative Processes
4. Direct Methods for Solving Sets of Linear Equations
5. The Fast Fourier Transform
6. Fast Multiplication of Numbers