

ible. Publishers appear to let the public decide on the quality by publishing most offerings—hence we have, in the reviewer's opinion, too many low quality books.

One would hope that higher quality books could be distinguished, perhaps like Open University recommended texts, but such hopes are little more than that. Authors are not being well served by the publishers who should review such texts with care; but the advent of modern desk-top publishing systems means that preparing a text that looks superficially good is easy.

The reviewer would like to see the approach taken by the mathematical texts produced under the pseudonym of N. Bourbaki in which a high consistent standard has been attained by a collaborative effort of several authors. Unfortunately, the practical difficulties abound. Moreover, professional bodies in many cases are becoming indistinguishable from other publishing houses in the standard of their work.

Returning to the book, and ignoring the many small errors, we have a text which presents software construction of re-usable components to a high standard using Ada. The strong point with the book, largely accounting for its length, is many worked examples for a student to study and emulate. As the name indicates, emphasis is (rightly) placed upon data abstraction which is aided by the Ada generic mechanism. Indeed, the book made one think how re-use could be effective in languages without a comparable mechanism.

However, the book could not replace conventional Ada textbooks, since it does not aim to teach the entire language—it is more of a source book of complete examples. At the price, it is good value for money, but a second edition would be hopefully be better still!

BRIAN WICHMANN  
*The Open University and NPL*

DOUGLAS A. YOUNG

*The X Window System: Programming and Applications with Xt (OSF/Motif Edition)*. Prentice-Hall, 1990, £33.50, 533 pp. softbound, ISBN 0-13-497074-8

The windowing system known as *X* is complex to program. The library of low-level routines initially used to implement software under *X*, and known as *Xlib*, is difficult to use and generates very large programs the bulk of which perform mundane message-passing tasks.

*Xt Intrinsics* provide a level of abstraction which remove from the programmer many of the more tedious tasks demanded by *Xlib*, and are provided as part of the standard *X* software distribution. For simplicity of programming, however, a *widget* set is desirable. The set known as *OSF/Motif* is implemented using *Xlib* and *Xt Intrinsics*, and straightforward programs using those widgets can be integrated with *Xlib* procedures for more complex applications.

This book describes how to program *X* applications

using the language *C*, the library *Xlib*, *Xt Intrinsics* and the *OSF/Motif* widget set.

It is not a book for the raw beginner—some experience with *X* as a user is necessary, as is good knowledge of *C*. The concepts behind *X* are first of all described, then sample programs are presented which illustrate to the reader to write simple *X* utilities. These include examples which use raw *Xlib*, a mixture of *Xlib* and *Xt Intrinsics*, finally integrating *OSF/Motif* widgets. Appendices are presented which clearly list the widgets and widget classes which *Xt* and *OSF/Motif* provide.

The style is clear, though fairly terse, and the pace rapid. For the serious programmer with access to an *X* terminal so that the examples can be experimented with, the book is excellent. Without a terminal, the reader will probably get lost quickly; it is not a book to browse through.

All the *X* programs presented are quite long, and to try them out requires a lot of typing, although copies of them can be obtained by anonymous FTP if your site is connected to the Internet. My only real concern is that where 'screendump' illustrations of windows are provided, they appear substantially different from those which appeared on my own screen. This is probably unavoidable, but certainly needs more discussion if the reader is to be reassured that the demonstration programs they have tried are in fact working correctly.

This is the best book I have yet seen which explains how to write programs under *X*. A second edition of the book is about to be published, and I am very much looking forward to reading it.

MIKE JOY  
*Warwick University*

TOM GILB and DOROTHY GRAHAM

*Software Inspection*. Addison-Wesley, 1993, £24.95, 472 pp. softbound, ISBN 0-201-63181-4

You may recall occasions on which you have 'burned'—discarded and rewritten—your programs, and perhaps even defended the practice before an incredulous project manager. Now you can quote Tom Gilb and Dorothy Graham at them. This book has to be the definitive text on software inspection. It covers history, costs and benefits, the process, the roles, and the difficulties; and it includes six informative and varied case studies contributed by other authors.

Gilb and Graham include in the process a causal analysis of defects, with feedback to improve the process itself. Good: processes are as error-prone as programs, and making methods themselves subject to systematic criticism avoids their all-too-frequent doctrinal rigidity. Oddly enough, it seems that inspection works better for other documents (requirements, for instance) than it does for programs. But why? The book is well written, albeit at some length, and surely everything that can be said about inspections is said; except *how they work* (in the