BOOKS

Books should be sent to the editor-in-chief. Selected books, which are within the scope of SCP and are not proceedings, will be reviewed. Others may be mentioned.

Programming System Methodologies. By Carol A. Ziegler. Prentice-Hall, Englewood Cliffs, NJ, 1983; ISBN 0-13-729905-2.

The strong feature of C. A. Ziegler's book is its breadth of coverage. The topics discussed go from system design to implementation issues; as such, the book is a good survey of current ideas on software methodology. This is one of the few works to include material on both 'industrial' methods (e.g. Warnier-Orr, Jackson, bubble diagrams)and more formal approaches (in particular abstract data types and formal program verification). Usually, the two schools listen (let alone talk) very little to each other; thus any attempt, such as Ziegler's to bridge the gap, is to be commended.

Nevertheless, the book suffers from several deficiencies.

First, depth seems to have been to often sacrificed for breadth. It may be a good idea to introduce topics such as cryptographic encryptlon into a book on system design (as part of the discussion on system security); one may doubt very much, however, whether the one and a half pages devoted by Ziegler to this topic will be very useful to the reader.

Second, the overall balance of topics is less than adequate. In the preface, the author laments that "standardization and design are rarely emphasized in programming classes" and that "the design of larger and more complex programs is usually neglected"; but the book suffers from exactly the same imbalance: only one fifth of the text (Chapters 2 and 3) really deals with high-level design; the rest is mostly concerned with detailed design and implementation. No requirements or specification method, formal or informal, is discussed. Another example of a missing topic is static program analysis, a practical and usable technique, which should have been introduced in the chapter on "program validation".

Third, there is a certain lack of perspective which sometimes makes the book look like a catalog. For example, Halstead's controversial 'software metrics' theory is discussed in some detail, but the author does not say how these techniques can fit, if at all, in a general method for software development.

Next, a number of imprecisions and slight misunderstandings are embarrassing. For example, the discussion on loop invariants (p. 213) is misleading. It is *not* true than an invariant "indicates the relationship of the values of the variables at one iteration to those of the preceding iteration so that induction can be used in the proof". Quite to the contrary, an invariant is a kind of 'absolute' property of the program variables which holds whenever control reaches a certain point; the beauty of this notion is that you don't have to explicitly relate the 'current' value to the 'previous' ones. In the book's only example using a loop invariant (a program which computes the gcd of two integers), the assertions refer to a variable's current value n, its previous value n', and its next-to-previous value n''. This, in my opinion, obscures the all-important notions of assertion and invariant (of course, there are theories in which assertions may involve both the current and initial states, e.g. [2], but Ziegler does not attempt to present such a theory in a clean way).

Next, the book follows a pattern, set by Kernighan and Plauger in their unpretentious but excellent little book [3] and also used by Ledgard [4] and more recently by Gries [1], of providing the reader with 'maxims' or 'proverbs' (e.g. here USE GOTOS SPARINGLY, etc.). This device may be useful if the maxims have been designed with great care (as e.g. in [3]), but easily turns into the motherhood and apple pie preaching otherwise. I very much doubt the relevance of a precept such as

DO NOT NEEDLESSLY TAMPER WITH A WORKING PROGRAM.

Such a piece of advice is probably harmless, but it is also useless: who in the world would advocate 'tampering' *needlessly* with a program (working or not)? What we want to know is *when* (if ever) one may consider modifying a working program! So here is a maxim for all maxim authors:

GIVE US PRACTICAL RULES, NOT PIOUS ADVICE; FOR EACH RULE,
SAY PRECISELY WHEN IT APPLIES AND WHEN IT DOES NOT.

My last criticism is less important and probably more subjective. It applies to the style of the book, which seems to have been rewritten by a technical editor according to the rules taught in some 'basic technical writing' courses offered in the USA. Thus most sentences are short and few depart from the 'subject-verb-complement' structure. When trying to pin down the reasons for my uneasiness with the style, I suddenly realized that I could not find any colon or semicolon! Why one should dispense altogether with these faithful servants of sentence articulation and reader's comfort is beyond my comprehension. Not being a native speaker of English, I had some qualms about including these comments in this review; my reason for not suppressing them is that such exaggerated terseness seems to reflect a fairly general trend in recent American textbooks. Some publishers may have gone overboard in trying to achieve simplicity in style. There *should* be some middle ground for technical style between Proust and BASIC.

Coming back to the book's technical contents, I do not agree that Ziegler's work is "primarily for computer science students rather than business students or the general programming public" (except from the Preface). I would not use it in a software engineering course because of its imprecision and lack of depth. It may, however, be useful to practicing programmers as a broad and readable tutorial on some of the current ideas on programming methodology.

Bertrand MEYER
University of California
Santa Barbara, U.S.A.

References

[1] D. Gries, *The Science of Programming* (Springer, Berlin, 1981).
[2] C.B. Jones, *Software Development: A Rigorous Approach* (Prentice-Hall, Englewood Cliffs, NJ, 1980).
[3] B.W. Kernighan and P.J. Plauger, *The Elements of Programming Style* (Addison-Wesley, Reading, MA, 1974).
[4] H.F. Ledgard, *Programming Proverbs* (Hayden, Rochelle Park, NJ, 1975).