References

BIRTWISTLE, G., DAHL, O-J., MYHRHAUG, B. and NYGAARD, K. (1973). *Simula Begin*, Student Litteratur, Auerbach.

BRINCH HANSEN, P. (1973). *Operating Systems Principles*, Prentice Hall, Englewood Cliffs, NJ.

HOARE, C. A. R. (1974). Monitors: an operating system structuring concept, *CACM*, Vol. 17 No. 10, pp. 549-557.

KAUBISCH, W. H., PERROTT, R. H. and HOARE, C. A. R. (1976). Quasiparallel Programming, *Software—Practice & Experience*, Vol. 6, pp. 341-356.

O'KANE, P. C. (1976). The Operation of a Hospital Paramedical Department—A Quantitative Study, Ph.D. Thesis, The Queen's University of Belfast, N. Ireland.

PERROTT, R. H. and RAJA, A. K. (1977). Quasiparallel Tracing, *Software—Practice & Experience*, Vol. 7, pp. 483-492.

WIRTH, N. (1971). The Programming Language PASCAL, *Acta Informatica*, Vol. 1, pp. 35-63.

Book reviews

Workshop on Reliable Software: Applied Computer Science by Peter Raulefs, 1979, 281 pages. *Carl Hanser Verlag*

This book presents the proceedings of the workshop on reliable software organised by the German Chapter of the ACM and held at Bonn University in September 1978. Twenty-two papers were selected for presentation and inclusion in the proceedings. Ten of these papers were submitted by researchers from Germany, six from the USA, and the rest from Italy, Austria, France and the UK. All but three of the papers appear in English in the book; for the sake of completeness, it is a pity that the remaining three were not also translated from German.

The general standard of presentation of individual papers is good but there is a paucity of introductory material, both to put the overall purpose of the workshop into perspective and to give some background on the major themes covered. There are no reports of panel discussions and audience responses so some of the possible benefits of publishing the proceedings of a workshop seem lost. However, the book is still of considerable interest because it contains a broad range of material devoted to several different aspects of the very important topic of software reliability. Papers are presented on program testing and fault detection; systems for program development, software design methodologies for industry, programming language concepts, specification and documentation, and program verification. Thus, there are papers with a theoretical emphasis—verification of while-programs in a simple calculus, for instance—and others with a more practical bent such as the application of new testing techniques to a numerical algorithms library. The book should be of interest to both the more theoretically inclined and to those with a practical concern in the production of reliable software.

BRIAN FORD (Oxford)

Reliable Software Through Composite Design by G. J. Myers; 1975; 159 pages. (*Van Nostrand Reinhold*, £5·20)

I am not sure why this book has only just come up for review—although the review notice stated that it was published in 1979, the book itself gives a 1975 date—perhaps having been produced in America it is only now being distributed in the UK. It is written by Glenford Myers who is already a well known authority on software production. He is the author of many research papers and a number of books and contributes to some of the 'up-market' seminars in this area.

This particular book of his describes what Myers calls 'composite design'. This is a design methodology which draws on many of the ideas often associated with Constantine, namely the functional decomposition type of approach, to be contrasted with the data structure methods of Jackson and others. Thus the book, after a good preamble in which 'program quality' is defined, deals first with the major attributes of modules—termed 'strength' for the internal relationships within a module, and 'coupling' for the links between modules. Other attributes of modules, such as size and predictability are discussed, and for each recommendations are given on what one should be aiming to achieve for a 'good piece of software'. The actual process of dividing a problem into modules is discussed under the heading of composite analysis using some good examples to illustrate the method. Concluding the part of the book specifically describing 'composite design' is a chapter which shows how it fits in with structured programming, documentation and other aspects of software production. There is then a chapter on modularity and virtual storage which I felt was a 'poor mans' version of his paper in one of the IBM journals. The book ends for me, however, on a very strong note with a 'model of program stability' in which the attributes of a piece of software are first quantified and then predictions can be made about the 'ease of change' of the software. Although some would see this venture as rather academic, the chapter does underline the reasons for aiming at good module design, and serves as a good conclusion and summary of much of the rest of the book.

The book is designed as a practical guide for experienced programmers and system analysts. It gives details of theories from the viewpoint of their use for both application program and system program design. I believe the book succeeds in these objectives: it is clear, easy to read and understand, and is not too long. Myers, almost apologetically, states that 'Composite Design is largely a collection of guidelines', and the book details these 'guidelines'—'not firm rules'. As for reliability, this is discussed briefly in the introduction and then it is assumed that the reader can 'see' that employing the methods of 'composite design' will lead to reliable software.

This is a useful guide for people engaged in the production of software systems on how to design the internal structure of a program.

DAVID JACOBS (Leatherhead)

Fortran, PL/I and the Algols, by Brian Meek, 1978; 291 pages. (Macmillan, £12·00)

This book is not an introductory programming text for the languages in its title, nor is it yet another book on the comparative study of programming languages. It would seem that the book falls nicely between two stools. Indeed that was the reviewer's opinion before reading the book. Brian Meek has in fact identified a completely empty stool and provided the appropriate book.

The book discusses six languages—ALGOL 60 (Revised and Modified), ALGOL 68, FORTRAN IV, FORTRAN 77 and PL/I—from the viewpoint of a user not the viewpoint of a 'language lawyer'. This provides a refreshing change. Indeed the book is very pleasant to read, being marred (for me) only by a galaxy of seemingly irrelevant quotations; although I must admit to liking one quotation from Aldous Huxley. Inevitably there are a few typographical errors, but none of them will cause the reader any problem. They will irritate the author more than the reader. I did not notice any errors in the language details. I do disagree with some remarks about Euclid's Algorithm and Ackermann's Function.

If you are involved with teaching a course on any of the languages or a course on a comparison of languages, or are simply interested in programming languages you should definitely read this book. The book will also make excellent background reading for a student on either course, but its choice of languages and its price suggest that it is not suitable for use as the text book for a course on the comparison of languages.

Regrettably the publishers chose to produce this book only in hardback; a paperback edition would surely have been at a price more attractive to students.

A. M. ADDYMAN (Manchester)