Books

Software Reliability, Principles and Practices, Glenford J. Myers, John Wiley and Sons, Inc., New York, New York, 1976. 353 pp. (ISBN 0-471-62765-8, $19.95).

Ever since the first load instruction was mistakenly coded in place of a load-address instruction, techniques to improve software reliability have been of major interest to programmers, analysts, managers, and most importantly, to general users of computer systems. Recently this interest has grown, probably because we now recognize that we have techniques available that increase the amount of code created per unit of work. (See, for example, the article by Walston and Felix in Volume 16, Number 1, 1977, of the *IBM Systems Journal.*) We now wish to use some of the programming productivity toward improving the reliability of the code that is created.

One example of this new emphasis on reliability has been an explosion in the amount of published material available. For example, both the September 1976 and December 1976 issues of the *ACM Computing Surveys* are devoted to reliable software. However, much of this published material is of little use to the individual who is trying to complete a software development project on time and within a budget. The techniques and subjects covered tend to be too theoretical or difficult to implement in the real-life world.

Software Reliability, Principles and Practices is the first general-purpose text that I have seen that is applicable to our everyday work of getting the system developed and into operation. After an initial section describing basic concepts of software reliability, the author covers the entire program development cycle, discussing techniques for each aspect of development from initial requirements statements to testing and acceptance. Checklists and suggestions are given where appropriate. I particularly liked the author's testing axioms and his checklist for use of high-level languages. A final section of the book discusses more advanced topics related to reliability.

A reader will not agree with all of the author's ideas and may even feel that some of his opinions and suggestions are not adequately supported. Also, the author may be criticized for describing a lot of useful items related to reliability without unity or organization. Since the development of reliable software is still not a science, I feel this is not a serious weakness.

I expect that many of the topics discussed by the author will become second nature to us within the next several years. Some will probably even be refuted by later research and experiences. However, the text does represent the best treatment of the current state of the art that I have seen. A development manager should make this book mandatory reading for all of his subordinates. It would be a wise investment, which would pay off in significantly less time being spent in debugging.

J. P. Langer
Hartford, CT

The editors assign reviews of books that might interest our readers. Reviews are signed, and opinions expressed are those of the reviewers.