Life Cycle Support in the Ada Environment

John McDermid and Knut Ripken
(Cambridge University Press, New York, 1984, 247 pp., $24.95)

More than two dozen full-length books have been written on various engineering and logistics aspects of the Ada programming language. Based on Pascal and other Algol-like languages, Ada is the first practical programming language to fully support the use of structured software engineering, data abstractions, concurrency, portability, encapsulation, generics, exception handling, and multitasking. It supports the development of affordable, quality software by addressing, among others, the critical issues of the programming environment, maintainability, security, and readability early on and up front.

Ada's software design capabilities far exceed those required for the effective design and productive coding of real-time embedded systems, formerly written in specialized languages such as Jovial or Coral 66. The programming community is expected to be much more receptive to Ada than to Tacpol and Jovial. Indeed, use of Ada may someday rival Fortran and Cobol in large software packages.

Much of the material for this book is derived from a study (of the same title) conducted for the Commission of the European Communities prior to publication of the ANSI/MIL-STD-1815A standard version of Ada on February 17, 1983. Many changes were made to Ada as a result of the ANSI standardization process. However, several of the methods and tools proposed by the authors (perhaps editors is a better word) to improve life-cycle support to large-scale software systems are mostly Ada independent. This independence is especially true for software management. Thus, the book is by no means obsolete.

The authors suggest guidelines for software development, maintenance, and management. They state that their aim is to produce a cohesive methodology to assist in the cost-effective production of reliable systems. Unfortunately, there is no mention of life-cycle cost modeling, which builds management confidence that a proposed large-scale software effort is affordable. Thus, an important function of program management—that of managing all resources including people and money—is largely ignored.

Nevertheless, the book provides a useful initial step toward developing the technical aspects of a coherent Ada programming support environment, or APSE. It goes far beyond the so-called Methodman published by the Department of Defense in 1982. Still, a modest design-to-cost emphasis for software managers would address the need to reduce major cost overruns on large software developments. Indeed, the best software managers are able to achieve a metastable balance among system performance, cost, schedule, and supportability.

This concise book, the first in the Ada Companion Series, contains a table of contents, four well-written chapters, five appendices, a list of more than 100 references, but no index. The chapters include such topics as an introduction to the requirements for, and the choice of a coherent APSE (discussion is on the life-cycle model, management methods, configuration control, system specification, and project management) and an outline of a coherent APSE (discussion is on development methods, system specifications, change processing, implementing an APSE, and support tools). Conclusions and recommendations are also given.

I found three crucial areas of software life-cycle support ignored or overlooked: software costing, software testing, and software training. Inadequate technical exchanges between design engineers and workers in these areas are likely to have significant adverse effects on the development or maintenance of quality software. Indeed, when testers, designers, and users do not or cannot talk to one another, the resulting disaster is comparable to that of the US Rangers during the recent Grenada operations. They could not communicate with supporting Navy personnel because their respective radios functioned on different frequencies.

In conclusion, the authors provide the first steps toward establishing guidelines for life-cycle support in the Ada environment but do not provide a totally integrated logistical support model. Since there are no student exercises or experiments to test understanding of the proposed methodologies, the book, by itself, is unsuitable for self-study or as a textbook. It is best used for supplementary reading by advanced undergraduates in computer science who are interested in project management and by managers responsible for the development, maintenance, and operation of large-scale software efforts.

Albert A. Mullin
USA BMD Systems Command
Huntsville, Alabama

November 1985