Recently published books and new periodically may be submitted for review to the book reviews editor:

Dr. Francis P. Mathur
Mathematics Department
California State Polytechnic University
3801 West Temple Avenue
Pomona, CA 91768
Telephone: (714) 598-4421

Note: Publications reviewed in this section are not available from the IEEE Computer Society; they must be ordered directly from the publisher. To request ordering information, circle the appropriate number on the Reader Service Card.

**Syntactic Pattern Recognition and Applications**—King-sun Fu (Prentice-Hall, Englewood Cliffs, N.J., 1982, 596 pp., $37.50)

This book, an expanded and updated version of King-sun Fu's 1974 work, *Syntactic Methods in Pattern Recognition*, uses the syntactic or structural approach to pattern recognition problems. The syntactic approach draws an analogy between pattern structure and language syntax or grammar. Patterns are specified as being built up from various compositions of subpatterns, just as words are built up by concatenating characters, and phrases and sentences are built up by concatenating words. For this approach to be advantageous, the simplest subpatterns—the pattern primitives—should be much easier to recognize than the patterns themselves. The determination of pattern primitives is covered briefly, but the main focus of the book is on the syntactic aspects.

A *pattern description language* provides the structural description of patterns in terms of a set of pattern primitives, and its so-called *grammar* specifies the rules governing the composition of primitives into patterns. After identification of each primitive within the pattern, recognition is accomplished by syntax analysis, or parsing of the *sentence* describing the given pattern to determine whether or not it is syntactically correct with respect to the specified grammar.

The author is a strong advocate of the syntactic approach. His self-contained book introduces formal languages and describes syntactic analysis without assuming prior background on the reader's part. He also includes a chapter on stochastic languages, whose grammars incorporate probabilities to deal with noisy or distorted patterns, and treats *grammatical inference*, the problem of learning a grammar based on a set of sample sentences. Two new chapters, one on applications to waveforms and contours and another on texture analysis, are of particular interest. The waveform applications include analyses of carotid pulse waves and EEGs.

Suitable for an upper-division or graduate course, this book could also be used by the practicing computer scientist for self-instruction. Although it lacks problem sets, it does contain extensive references, and a series of appendices provide detailed grammars for patterns ranging from Chinese and Roman characters to the texture of a reptile's skin.

Oscar Firschein
Lockheed, Palo Alto
Reader Service Number 98

**Software Design: Methods and Techniques**—Lawrence J. Peters (Yourdon Press, New York, 1981, 234 pp., $24.00)

The rapidly escalating costs of software development have resulted in a large number of techniques for organizing and evaluating the software design process. Unfortunately, many of these techniques have been developed independently by software industries to suit their special needs, with little joint effort devoted to developing a comprehensive software design methodology. The result is a proliferation of diverse techniques, with little help available to the designer who needs to make an informed selection. While Peters attempts to fill this need, the book is not intended as a compendium of all major design techniques; rather, the author provides only a brief overview of the techniques and presents a comparative summary to give designers a quick idea of which methods they need to explore further. In this regard, the book remains unique despite the recent avalanche of material on software design issues.

The book is divided into four parts. Part 1 explains what software engineering and software design are. A morphological model of software engineering and various software life cycle models are presented first. The book then attempts to define the design process from a number of different viewpoints.

Part 2 presents several design representation techniques, which are classified into four categories: architectural, structural, behavioral, and database. The distinction between the first two categories is somewhat fuzzy and combining them into one may have been a better approach. In the structural category, the discussion on SADT is too brief to make
a novice feel comfortable with it. The exposition in Part 2 is generally excellent except that some techniques need more detail. A comparative summary of all methods in the first three categories is most rewarding; I wonder why such a summary was omitted from the discussion on database structure.

Part 3 discusses software design methods, which are divided into three categories: data-flow oriented, data-structure oriented, and prescriptive. The last category includes several less well-known or special-purpose techniques, which have not been developed into a full-fledged design method. As in Part 2, the discussion is generally excellent and comparative summaries have been provided; however, adequate discussion of examples seems to be lacking for some techniques.

Part 4 is concerned with the overall issue of planning and executing a large software development project. A morphological framework for evaluating the overall design has been proposed. The book concludes with an overview of the most pressing research problems in software engineering.

As expected, the book does emphasize the need for good design and the separation between design and implementation. However, the point could have been made more convincingly by providing an actual example showing the problems that arise when one fails to consider these requirements. (For more on this topic see G. D. Bergland’s “A Guided Tour of Program Design Methodologies,” Computer, Vol. 14, No. 10, Oct. 1981.) In the examples, nonsense symbols are used to denote various items in the diagrams when elucidating a concept. This is very distracting; the use of real names based on some well-defined problem would have greatly enhanced the utility of the diagrams and obviated some of the actual examples. Several typographical errors were also noted, mostly in the form of missing characters and words.

Overall, the author has done a commendable job, and the book does largely meet its stated goals. It should prove invaluable to software professionals who are faced with the problem of selecting a design methodology that best suits their needs. As the author notes, this may very well be a unique combination of several of the methodologies discussed in the book. The book should also be of great help to a novice who simply wants to get a glimpse of what is currently available in software design technology.

Krishna Kant
Northwestern University
Reader Service Number 99

The BASIC Handbook—David A. Lieu (CompuSoft Publishing, San Diego, 1981, 480 pp., $19.95)

Some Basic users may wish to rewrite a program that uses a computer different from their own. Where do they turn for an encyclopedia of Basic words? The BASIC Handbook is a definite possibility.

Written by an author who is both an educator and a practicing engineer, the book is the second edition of a handbook on Basic language text for 250 computers. It introduces 283 additional words, bringing the total to almost 500. Its primary purpose is to provide Basic language information they can apply towards solving the program incompatibility that arises from Basic dialects or words not shared by the computers. Basic words to control peripheral devices such as disks, tapes, and printers are excluded from the handbook, since Lieu believes they show little uniformity.

The greatest potential of this single-volume handbook seems to be for users who desire to run a Basic program written for one computer on a different one—usually their own. However, not every program can be successfully converted and programs that can run on every computer are rare. The toughest conversion problems involve programs from color video game machines that include a “custom” Basic interpreter and undocumented manufacturers’ glitches. Also troublesome are the programs that rely heavily on PEEKs and POKEs. Translation of these programs is nearly impossible without memory maps of computers plus the knowledge of how to use the maps—both of which require the skills of a highly experienced programmer.

To make the most out of the handbook, Lieu recommends the following steps: (1) break the original program into small functional blocks or modules, (2) include REMark statements liberally throughout easy documentation, and (3) determine how to accomplish the goal of each block by using the Basic words and features that are part of your own computer.

Information about each Basic word is divided into eleven parts. The first part is the word itself. If “ANSI” appears in the second part, it means the word is part of the NBS American National Standard Institute’s minimum vocabulary. The third part identifies the word as a function, command or statement and is followed by a word description. The next three parts illustrate (1) a program to test if the computer recognizes the word, (2) expected response to test the program, and (3) helpful hints on programming.

When different spellings are used on different computers they are noted in the part on “Alternate Spelling.” The part on “If Your Computer Doesn’t Have It” gives alternative ways to accomplish the same purpose by using other Basic words when possible. The last two parts cover variations in word usage and reference other words for more detail.

The author also discusses special Acorn, Atari, Tektronics, and TRS-80 color features for handbook users. He then provides a brief summary treatment of common Disk Basic features. The summary aids users in rewriting the program by helping them recognize when Disk Basic is being used and what it is used for. (For more information of a tutorial nature, the author recommends CompuSoft Publishing’s books on Disk Basic and DOS.)

Judy M. Myerson
Philadelphia, Pa.

Reader Service Number 100

Use order form on p. 128C

Contains 34 papers presenting a multi-faceted view of the ideas and processes associated with the current practices of systems analysis and design. Formerly priced at $67.53 pp.

Order #454

Proceedings—Workshop on Systems Analysis and Design—A Foundation for the 1980’s

Atlanta, GA

Members—$21.00
Nonmembers—$29.00