Books

Object-Oriented Programming: An Evolutionary Approach, Brad J. Cox, Addison-Wesley Publishing Company, Reading, MA, 1987. 274 pp. (ISBN 0-201-10393-1).

The term *object oriented* is the victim of hyperbole. Since the ideas attracted favorable attention in the early 1980s, the term has been distorted by proponents of particular technologies and products. Potential readers might pass by the Cox book with hardly a glance. They shouldn't. Cox identifies precisely what he means by object oriented, the programming style pioneered by Smalltalk. He describes the concepts lucidly and relates them to practical programming in a way this reviewer has not previously encountered.

The book declares its objective in the preface: "Object-oriented programming . . . puts reusability at the center of the software development process, making reusability the usual way that new components are built, instead of occasionally calling a function from some library as programmers now do. . . . It will be described thoroughly enough to show how hybrids can be built on [conventional] base languages." The book then, chapter by chapter, never strays; often the connection between each immediate topic and the objective is explicitly drawn.

The practical thrust of the book is exemplified by the discussion in chapter 6 of the social environment required to achieve reuse. The analogy with integrated circuits is useful to introduce the topic, but the term *Software-IC* becomes tedious as a paraphrase for "encapsulated type definition." Another practical compromise Cox emphasizes is the ability "to bypass the object-oriented machinery to access an object's private information directly. This is one of a hybrid language's greatest theoretical weaknesses and one of its greatest pragmatic strengths."

Implementations of object-oriented languages have mostly been embedded in operating environments. The main advantage over older programming techniques, such as reusability, is often lost because of operating system dependencies. This has slowed commercial applications. Using a proprietary language—Objective-C™—for his examples, Cox separates the language from the environment.

The first principal tool, "encapsulation . . . restricting the effects of change," is repeatedly illustrated. A concise example, managing the capacity of an "Ordered Collection" (chapter 8), is persuasive.

The second principal tool, "inheritance . . . for automatically broadcasting code to classes developed by different members of a team," is most persuasively illustrated in chapter 7. Passivation and activation serve to translate executable versions of objects to and from linear representations, providing for checkpoint/recovery and transmission from one application to another. A single implementation in the ultimate ancestor class, *Object*, suffices.

Most of what is published about object-oriented programming is directed toward personal computing services. Its focus on productivity of programmers who are more concerned with innovation and improving their own tools can be traced to the Xerox PARC Learning Research group in the late 1970s. Commercial programming and the work of large development teams have not received proportionate attention. Cox starts to remedy the imbalance, but does not go as far as he might. He pays more attention than other writers to programming standards and the control of user interface appearance. However, his quantitative examples address the efficiency of machines rather than the effectiveness of programmer teams.

Cox's largest programming example, "Iconic User Interfaces" in chapter 9, is an excellent illustration not only of the fact that programming is still a complex task, but also of the way in which window interfaces are constructed.

However, many expositions illustrate encapsulation and inheritance by building object-oriented screen interfaces with windows, icons, and pop-up menus. This distracts the student from the language features that should be his or her focus. This reviewer would have preferred an example from some other area, such as telecommunications control, scientific programming, or computer-aided design of "hardware ICs."

Cox articulates a keen appreciation of the limits of his topic, especially in chapter 10, "Different Tools for Different Jobs," where he separates what is essential to object-oriented programming and what can be discussed under other labels. The technical focus is on storage management and addressing scope, addressed as engineering decisions. What is best for long-running interactive applications is different from what is best for batch applications; alternative designs are briefly discussed. The chapter finishes with a deserved indictment of another subject of hyperbole: "Although system builders often speak of office automation systems, such systems hardly even address office productivity at all, let alone automate it. . . . A true office automation system has to be able to carry . . . binding agreements. . . . understanding for words like role, individual, responsibility, and concurrency, and recognize that they embody what is truly important . . . . Is object-oriented programming the answer? Absolutely not. It merely helps develop large, complicated, but basically conventional systems."

Style, layout, and typography are all good. The presentation is clear, to the point, and relatively complete. The programming examples are developed to the right level of detail to illustrate the points intended, and, in chapter 9, to suggest what an application prototype would require. One disappointment is the index, which contains inaccuracies. For example, the term *identifiers* refers to page 234, where we find the specification of the *Object class*, but not the word "identifier"; the mystery might be explained on page 54, where the author states, "Objects are identified by a new Objective-C data type called an id, or object identifier," but we cannot be quite sure.

Another example is "portability of Objective-C," with which this reviewer sought to check his own suppositions against author's claims. The reviewer never did find the discussion of portability he sought. Objective-C applications seem to be as portable as the underlying C language, since the language is implemented as C-macros and a discipline on link libraries. Personal experience is needed to test the limits of portability and reusability.

The appeal of Cox's book is that it shows specifically how an important methodology may help. After reading *Object-Oriented Programming: An Evolutionary Approach*, many readers will want to try the methods on a real problem. Most books leave this reviewer feeling that he either understands everything offered well before the end of the book, or is exhausted by more than he can absorb; this book is the exception.

H. M. Gladney
IBM Almaden Research Center
San Jose
California

Objective C is a trademark of Stepstone, Inc.

File Structures: A Conceptional Tool Kit, Michael J. Folk and Bill Zoellick, Addison-Wesley Publishing Company, Reading, MA, 1987. 528 pp. (ISBN 0-201-12003-8).

This book discusses the implementation of file structures on secondary storage, beginning with simple sequential files and building up—via sorting, merging, and searching—to B-trees and hashing. On the way, it discusses the basic features of disks, tapes, and buffering.

The text is very clearly written, well structured, and goes into ample detail. The many excellent examples illustrate both principles and applications. If the reader finds too much detail, the organization of the book makes it easy to skip material. Probably not very many people will read every word cover to cover, but all should be able to find what they need in this useful introduction and reference work. It is also a source of code that one may freely adapt to one's own purposes.

My main reservation is the book's limited scope. It covers neither the problems of linking collections of data of different types nor the physical

interleaving of different record types, which are central to the implementation of database systems such as DBTG, IMS, and DB2. The book is correct to avoid the higher-level aspects of databases. However, it describes file structures as "an application of data-structure techniques to . . . data on secondary storage devices." Does this include database physical storage?

These limitations are probably derived from Course CS-5 in the *ACM Curriculum '78* and, although they do not detract from the book as teaching material, tend to make it less attractive for a professional reader.

Overall, the excellence of the book in the areas covered makes it well worth reading.

Stephen Todd
IBM United Kingdom
Scientific Centre
Winchester, Hants

OS/2™: A Business Perspective, Dick Conklin, John Wiley & Sons, Inc., New York, 1988. 258 pp. (ISBN 0-471-63503-0).

True to its title, Dick Conklin's, *OS/2™: A Business Perspective*, gives the IBM PC user a broad perspective of OS/2 in the context of MS-DOS and the IBM family of personal computers. Emanating from classes and seminars delivered to varied audiences, the book reflects the kind of information a decision maker needs before embarking on a road to OS/2. Although programmers and technicians will benefit from the descriptive nature of the chapters, the level of the book targets it to managers and users.

As one introductory section title suggests, the author sees OS/2 as a bridge between DOS and System Application Architecture (SAA). This viewpoint pervades the early chapters and gives the reader an understanding and appreciation for the evolution of the IBM PC, PS/2®, and OS/2. The relationship between hardware and operating system is emphasized with informative descriptions of the Intel® chip family (8088 through the 80386), IBM PC family (original IBM PC through the IBM PS/2 Model 80), and DOS-OS/2 development (DOS 1.0 through DOS 3.3, OS/2 Standard Edition, OS/2 Extended Edition, and finally OS/2 and its relationship to SAA).

Chapter 1 is introductory in nature and describes the basic concepts of OS/2 and the role of an operating system as an interface between the bare machine and the applications. Chapter 2 discusses the first major motivation for a new operating system to replace DOS, taking advantage of the new microprocessor capability for addressing larger memory. This chapter provides a good description of how memory is addressed on earlier PCs and why there is a 640K DOS limit. Even experienced users will appreciate the discussion on breaking this barrier and the motivation for chip and operating system design which can take advantage of larger memory.

In chapter 3, multitasking, the second motivation for a new operating system, is presented. The need for OS/2 is presented against the background of what came before: DOS shells, memory resident "pop-up" programs, printer spooling, and multitasking windowed environments such as IBM's TopView® and Microsoft's Windows®. A description of the Intel 80286/80386 hardware support for multitasking describes the *real* and *protect* modes (what IBM calls DOS and OS/2 modes, respectively) and how segments are managed. There is also a section on security and the protection-ring structure of the 80286. The chapter concludes with a discussion of the OS/2 multitasking user and programming interface, although entire chapters are devoted to these topics later in the book.

Chapter 4 describes the user interface to OS/2. This includes a discussion of PC-DOS for comparison and the PS/2 user interface components (keyboard, monitor, and mouse). The first part of the chapter describes the single screen, special key stroke user interface for OS/2 Standard Edition 1.0. It describes how to boot the system, start new programs, and switch between them. The special key sequences to perform some of these functions (e.g., <alt esc> to rotate between running programs) and some for the OS/2 commands are presented here. The latter part of the chapter describes the graphics and windowed interface of OS/2 Standard Edition 1.1 which is the OS/2 Presentation Manager™ and conforms to the Common User Access part of SAA.

The issues involved in converting from DOS to OS/2 are covered in chapters 5 and 6. Chapter 5 sets the stage by describing the DOS 3.3 environment and reviewing the existing hardware and software support for larger memories and multitasking. It then explains how to migrate data and applications from the DOS to OS/2. Chapter 6 details the issues involved in installing and maintaining an OS/2 system, includ-

ing application installation, disk management, configuration parameters, and OS/2 performance.

The final chapters of the book take the reader in two different directions. Chapter 7 goes beyond the OS/2 Standard Edition to the OS/2 Extended Edition which includes two IBM products, the Communication Manager™ and the Database Manager™. This chapter tries to paint a total picture of where IBM OS/2 fits in the move toward an integrated SAA world. The last chapter turns inward and discusses some of the technical aspects of the OS/2 programming interface. Conklin explores the system interface and programming aspects for multitasking and resource management. A brief description of the OS/2 Application Program Interface (API) is presented with a list of the different system calls available to the application developer.

It is difficult to write a technical book in layman's terms. Conklin has tried to give the reader an understanding of OS/2, its motivation, and why one would want to migrate to a more sophisticated operating system. Through ample diagrams, prechapter glossaries, and a perspective which respects and reinforces the existing knowledge base of the reader, the author has written an enjoyable and informative introduction to OS/2.

Daniel Farkas
Assistant Professor
Pace University
White Plains
New York

Expert Systems Applications to Telecommunications, Jay Liebowitz, Ed., John Wiley & Sons, Inc., New York, 1988. 371 pp. (ISBN 0-471-62459-4).

It is universally recognized that modern telecommunications networks are increasing in complexity at an astounding rate. This complexity derives both from the sheer number of resources that make up these networks and from the diversity of equipment and the variety of communications protocols that may be incorporated in a single network. The level of knowledge and sophistication required of the people who must design, operate, and administer such networks has grown at nearly the same pace. These factors, coupled with the fact that expert systems have emerged among the first serious commercial successes of research in Artificial Intelligence, have made the topic of applying expert systems to problems in the telecommunications field one of the hottest at industry conferences, in technical journals, and in more popular trade publications.

It is very difficult to conceive of a textbook that could adequately treat an area that is still so early in its development. There are numerous research and prototyping efforts underway throughout the data processing and telecommunications communities to find ways to exploit expert systems technology. Nevertheless, this book offers an interesting collection of viewpoints on the subject. In a field where there is not yet a well-established repertoire of tricks and techniques, this book provides a glimpse of a variety of application development programs at different stages of maturity.

The book is like a collection of papers from a workshop on expert systems in telecommunications. It is divided into three sections, Case Studies, Methodologies, and Future Applications, with the case studies making up over two-thirds of the text. Each chapter in the Case Studies section reflects the views and experiences of a different group of developers, with the telephone industry and programs sponsored by the Federal government being best represented. As the editor notes in his introduction, some very significant and successful projects are not included in this selection of case studies, but it is difficult to deny that telephone companies and government agencies have been among the most visible proponents of this technology.

The case studies that provide the most self-conscious descriptions of the process of developing expert systems applications are those describing FIS, a project of the U.S. Navy Center for Applied Research in Artificial Intelligence; FIESTA, a NASA-sponsored project; and XTEL, a project initiated to cope with requirements of the Defense Communications Agency.

FIS, a Fault Isolation System, uses descriptions of the causal relationships between failures and measurable parameters to assist in locating problems in elec-

tronic equipment. The case study describes some of the issues that were addressed in building the knowledge acquisition aids for the project, and outlines the reasoning schemes that were developed. FIESTA is a Fault Isolation Expert System for TDRSS Applications and was created to assist operators in the NASA Network Control Center responsible for managing the Tracking and Data Relay Satellite System network. The case study for this system outlines the types of problems that face NASA operators, and the expert system is portrayed as being integrated into the existing network management structure. The kinds of knowledge about network components and processes that are represented in the system and the scheme used to manipulate certainty factors also receive special attention. XTEL is a design tool built to assist in planning the types of military telecommunications that are needed to meet the requirements of a regional conflict. The case study focuses on the kinds of knowledge about the characteristics and vulnerabilities of various media and frequency ranges that are required to allow the system to imitate experienced network designers and spends considerable time on the problems associated with presenting a flexible user interface.

The section on Methodologies contains some serious thoughts about the relationship between building expert systems and building traditional applications. Although there are no prescriptions for success here, the term *expert system* has had a sort of mystical cachet that hides the amount of hard work required to build and field such a system. The set of tasks that the system will perform must be carefully selected, the performance and interface requirements that the system must meet have to be well documented and understood, and then an effective means must be devised for verifying that the system does indeed meet the requirements and perform the designated tasks. It is helpful to see a thoughtful discussion of these kinds of problems published.

The section on Future Applications is not so much a projection of the eventual impact of expert systems on telecommunications as a place for articles on projects which had not yet reached the prototype phase when the book went to press. On the other hand, many of the case study chapters include paragraphs on the directions those projects were anticipated to take in the future, and general predictions about the future of a relatively new and evolving set of techniques are seldom very reliable.

The book is not without weakness. It is a loosely edited collection of papers and suffers from a certain amount of repetitiveness. Many of the authors address the question of why expert systems are attractive as solutions to problems in telecommunications, and many reiterate the standard sorts of lists of what is required for an expert system application to be successful. The individual chapters are not well integrated, and there is significant variation in the level of detail from one chapter to the next. Some of the systems described have already been treated at length in other books and in conference proceedings. The fact that the chapters are not better coordinated detracts from the book's value as a text, since there are no unifying themes and the authors of later chapters are unable to refer back to results or observations appearing earlier in the book.

Still, *Expert Systems Applications to Telecommunications* does offer an interesting sampling of work in this area, emphasizing a very practical perspective. In a field where it is often difficult to distinguish substance from optimistic speculation, this is a useful contribution.

Stephen Brady
IBM Thomas J. Watson
Research Center
Yorktown Heights
New York

On Writing Well, 3rd edition, William Zinsser, Harper and Row Publishers, New York, 1985. 246 pp. (ISBN 0-06-015409-8). *Writing to Learn*, William Zinsser, Harper and Row Publishers, New York, 1988. 256 pp. (ISBN 0-06-015884-0).

Written and oral communication are vital skills for those of us in technical fields. We must present ideas in a way that is understandable and interesting. William Zinsser offers two entertaining and well-written books that address how we can improve our writing skills.

I loved reading these books, because their messages are useful and the presentation makes them fun to read. Zinsser follows his own advice by taking the time to write simply and clearly.

On Writing Well should be read by anyone who writes anything from memos to books. Whatever a writer's message, the goal is to get the reader to understand its meaning. Zinsser points out how sim-

ple ideas can be made unnecessarily complex when they are written. He gives concrete suggestions on how to achieve simplicity and cohesion in writing. His principles also apply to anyone who teaches or does public speaking.

I appreciate writers and speakers who have taken time to prepare their presentation so that they get to the point quickly. Zinsser does this, presenting useful information in such a way that reading this book makes good use of the reader's time.

The author discusses how to begin writing, how to let one idea lead to the next, and how to finish. He stresses the importance of maintaining momentum through the writing and keeping the reader's interest. All of the ideas are simple, but until now no one has conveyed them to me with such clarity.

This is not a book that teaches how to write correctly. It is a book that teaches how to write simply, with a style that holds the reader's interest. If I had read this book earlier in my life, I wouldn't have been so afraid of using a sense of myself, my beliefs, or my own experience in writing.

When I finished reading *On Writing Well*, I had been given lots of ideas on how to improve my writing and had been entertained in the process.

Writing to Learn is an excellent book to read after *On Writing Well*. Zinsser continues in this book by saying that if we can think clearly about a subject and understand it well, we should be able to express ourselves clearly when writing about it. If we don't fully understand a subject, Zinsser feels that we can learn through the process of writing itself.

I believe that *Writing to Learn* would be of interest to people in technical areas. It may seem inconceivable that writing in some subject areas can be made both interesting and understandable. However, this book shows that it can be done and encourages more of us to do the same.

This book is full of examples by authors who know their subjects well and can explain them in simple terms. Zinsser includes examples of good writing in fields such as mathematics, science, chemistry, geology, and art. One of his prime examples is from Albert Einstein's book on relativity. Einstein's writing is clear and understandable without a suggestion of talking "down" to his audience. His style is given as an illustration that even the most complex concepts can be explained clearly. "If clear writing is clear thinking," Zinsser explains, "a mind clear enough to think of the theory of relativity would be likely to express itself simply and well."

My favorite part of this book is where the author states that no one should "have to read a sentence . . . twice to find out what it means." At one time I believed that I was a poor reader, since I often had to read sentences more than once; now I suspect that I may have been reading poorly prepared writing.

Sharon Perkins
Assistant Professor
University of Houston
Clear Lake
Texas