Understandable and Usable OO Textbook

George B. Durham, Boeing

Object-Oriented Software Design and Construction with C++ *by Dennis Kafura, Prentice Hall, Upper Saddle River, New Jersey, 1998, ISBN 0-13-901349-0, 440 pp., $44.*

Object-Oriented Software Design and Construction with C++ is a well-written introduction to object-oriented technology and would make an excellent textbook for an introductory course. Kafura covers the basic OO concepts in easily understood terms, providing everyday examples, and he fully explains all references. The only problem I found was the inclusion of Chapter 5, covering program compiling and debugging, which seems out of place in this book.

The author develops the book by presenting a high-level view of OO concepts, which he then breaks down into small object examples. He explains each object completely, giving alternatives when appropriate. Following the OO methodology, Kafura then explains how to link, or combine, these objects in terms of larger objects. He also clearly explains associations, aggregation, and inheritance. Note that this is an OO design book. The author has chosen to use C++ for the examples, but this is not a C++ tutorial. If you need to learn C++, go elsewhere.

One of the most refreshing parts of this book is that the examples are not only clearly defined and fully explained, but they are also usable examples rather than just programming tokens. The plentiful and purposeful exercises add greatly to the reader's understanding. Although, at first glance, several of the exercises appear to be too similar, there are subtle changes that ensure complete coverage of the subject. All students, either in the classroom or on their own, will appreciate the development of the exercises from general and relatively simple to sophisticated and challenging.

This is a "Web-enhanced" book. In this case, that means that the book is online at the publisher's site. The online version has a few more exercises in multiple formats, such as multiple choice, true or false, or essay. These exercises can be submitted for grading. The online version also includes Java applets, which allows the student to see what the author's version of the code looks like in action.

The main weakness of this book, in my opinion, is that it is missing a section on the Unified Modeling Language. The discussion of OO design should have included UML, since it is rapidly becoming, if it isn't already, the standard. Nevertheless, if you are looking for a well-written introduction to OO design, you should pick up a copy of this book, even if you are not a C++ fan. ❖

EDITOR: Michael Yacci • Rochester Institute of Technology • may@it.rit.edu

Decrypting the Past and Exploring the Future

Diomidis Spinellis, University of the Aegean

Decrypted Secrets: Methods and Maxims of Cryptology *by Friedrich L. Bauer, Springer Verlag, New York, 1997, ISBN 3-540-60418-9, 450 pp., $39.95.*

Cryptology provides the foundations for information security and privacy in our global Internetworked society. *Decrypted Secrets* succeeds in blending the history, applications, theory, and practice of cryptography and cryptanalysis in a readable and interesting mix. The book's content is equally divided between cryptography and cryptanalysis; it can almost serve as a self-contained cryptology reference.

A Great Reference

The first half of the book contains an exposition of various encryption methods. Bauer illustrates many methods with real-life historical examples. The description of mechanical encryption devices, such as Enigma, is particularly detailed. In a short section

0740-7459/99/$10.00 © 1999 IEEE

at the end of the first part, Bauer introduces some cryptology maxims that authoritatively explain a number of cryptological principles that are often anecdotally presented.

The second half of the book covers cryptanalysis. Most of the methods Bauer covers are based on linguistic properties of the unencrypted text—cleartext. Those who have read the history of the Enigma code breaking during World War II will be fascinated by a case-by-case description of the theory and practice behind those efforts.

The most important asset of the book is that it really does cover cryptanalysis. Using numerous examples, the author guides us through increasingly sophisticated methods of cryptanalysis, beginning with exhaustive combinatorial searches and ending with periodicity and alignment examinations. Although Bauer presents the theoretical foundations of the methods in full detail, readers who are not mathematically inclined can skip the theorem proofs and follow the detailed, step-by-step examples. Having been spoiled by the increasingly clever cryptanalytic attacks of the cryptographic methods presented in the first half of the book, I was almost expecting to see modern cryptographic algorithms like DES and IDEA in the second half. Unfortunately, there is almost no coverage of modern cipher systems, except for a description of possible cryptanalytic attacks on RSA and a brief mention of differential cryptanalysis.

I commend the author and the publisher for producing a truly beautiful book. The cover, binding, and typesetting are superb. What clearly sets the book apart are the wonderful and rare illustrations and the color plates of cipher texts, encryption machines, and operating manuals. In addition, although he covers it lightly, Bauer refreshingly discusses the policy debate on cryptography from a non-US-centric perspective.

But Nobody's Perfect

Despite the many commendable qualities of the book, some minor problems spoil its overall utility. The almost total lack of references behind factual and theoretical descriptions is the most important. The wealth of information in the book means that references could easily double its size, but given the scarcity of other resources it would be well worth it. Although most illustrations are interesting and well reproduced, the DES drawings could have been typeset better and labeled in English. Similarly, many fictitious examples should have been phrased in English instead of German. Cryptographic protocols, hash functions, digital hardware, software, and digital signatures are also very lightly covered. Finally, the flow of text is sometimes choppy. Some paragraphs abruptly begin explaining a theoretical concept without any obvious leads from the previous section.

Decrypted Secrets is a unique book. Because it covers in depth all methods of historical interest, it's a great companion to *The Codebreakers* by David Kahn (Scribner, 1996), providing the theory behind Kahn's narrative exposition. It also provides historical context and cryptanalytical background to *Applied Cryptography* by Bruce Schneier (John Wiley & Sons, 1996). One of the maxims that Bauer presents is Kerckhoff's famous "only a cryptanalyst can judge the security of a crypto system … The combined wealth of cryptographic and cryptanalytic information available in a single volume can guide us toward more secure systems." ❖

Software Engineering—A Broad Picture of the Discipline

Mariá Bieliková, Slovak University of Technology, Bratislava

Software Engineering: Theory and Practice *by Shari Lawrence Pfleeger, Prentice Hall, Upper Saddle River, New Jersey, 1998, ISBN 0-13-624842-X, 576 pp., $66.*

The complex nature of software development requires even simple systems to use principles of engineering in their development. In the last few decades, software development has become more and more an engineering activity, although it is still far from being a standard discipline, such as mechanical engineering.

Related textbooks are an important factor in the

growth of software engineering. Presently, several well-known software engineering textbooks exist: Ian Sommerville's *Software Engineering* (1996), Roger Pressman's *Software Engineering: A Practicioner's Approach* (1996), Stephen Schach's *Software Engineering* (1993), Pankaj Jalote's *An Integrated Approach to Software Engineering* (1997), and J.C. Van Vliet's *Software Engineering* (1993).

Pfleeger's *Software Engineering* is another such textbook. However, many distinctive features make Pfleeger's not just another book about software engineering.

Pfleeger's book blends project management and software management processes with the software development process.

Processes Explained

The methods, tools, and techniques described in each of the above mentioned books are quite similar. However, each book has a unique design. The method each uses to explain software engineering concepts is influenced predominantly by the extent to which each uses examples and describes the software engineering processes.

Conventionally, software engineering textbooks focus mainly on the development process. Most consider project, software, and process management processes as separate issues and limit their scope. Pfleeger's book blends project management and software management processes with the software development process. In particular, measurement issues are considered an integral part of software development, rather than as a separate discipline. Similar efforts are visible in Jalote's *Software Engineering*.

The Book's Structure

Pfleeger's book is organized into a preface, 12 chapters, a bibliography, and an index. Most of the items in the bibliography are briefly annotated. References to any other software engineering textbooks are missing.

Each chapter includes an illustrative application of the concepts described in the chapter. Applications relate to some aspects of two examples: a typical information system (a system to sell advertising time for Piccadilly Television) and a real-time system (based on the embedded software in the Ariane-5, a space rocket belonging to the European Space Agency).

Each chapter also provides an interesting summary. The results at the end of each chapter are expressed in three ways: what the chapter's content means to an individual developer, to a development team, and to researchers. Finally, it is worth noting that each chapter contains a list of key references and exercises.

The chapters can logically be distributed into three parts, although this is not indicated in the Table of Contents. The first part, Chapters 1–3, explains why knowing software engineering concepts is important. It discusses the need for understanding process issues and for doing careful project planning.

Part Two, Chapters 4–10, walks through software development process activities, regardless of the process model used to develop the software. Besides the activities described in most of the textbooks, Pfleeger does not exclude topics linked to implementation—for example, program writing. Some recent software engineering textbooks don't consider this topic, such as Ian Sommerville's and Roger Pressman's. Pfleeger explains the role of programming standards, describing the structure of a program's documentation. She also presents several programming guidelines related to three major aspects of each program: control structures, algorithms, and data structures. Moreover, the chapter devoted to delivering the system is unique, a subject often not addressed in similar textbooks.

Pfleeger discusses the activities of the software life cycle in the context of conventional and innovative life cycles. She presents methods used in software engineering, such as object-oriented development, structured development, and formal methods.

Part Three, Chapters 11–12, focuses on some aspects of process management—evaluating products, processes, and resources, as well as improving predictions, products, processes, and resources.

The book begins and ends with Wasserman's eight concepts (see *IEEE Software*, Nov. 1996, pp. 23–31). Together, these constitute a viable foundation for a discipline of software engineering.[6] Between the concepts lies a compelling discussion of the future of software engineering by means of examining current software engineering practices.

Research vs. Practice

Software engineering research has made tremen-

dous strides over the past years but has unfortunately had little affect on industrial practices. Pfleeger's aim was to contribute to the process of transferring software engineering research into everyday use.

The book complements theoretical foundations of software engineering through sidebars with case studies, and examples of its use in successful and unsuccessful projects. This provides insight into various project situations. Moreover, the two examples based on actual projects illustrate the explained concepts, even though one of them is an example of an unsuccessful software project.

Pfleeger's book gives a broad picture of the software engineering field. However, with so many concepts, methods, tools, and management views in software engineering, it is impossible to cover everything in only 500 pages. Readers—above all undergraduate students—will need to exercise considerable judgment in applying its insights to their particular projects. Blending practice and theory with an up-to-date bibliography can help.

One of the book's helpful features is that it has a manual—C.B. Seaman's *Solutions to Exercises*, Prentice Hall, 1998—which gives some solutions to all the exercises in the book (many of the exercises include open-ended questions and do not lend themselves to definite solutions). Perhaps due to the book's wide scope, the exercises don't usually address specific details. Only a few of them come with described methods, techniques, and tools. Many of the questions are related to legal, ethical, and moral issues.

Readers can further update their knowledge by studying additional materials provided online (www.prenhall.com). Each chapter of the book has a corresponding section containing links to examples of existing projects, tools, articles, and books with a wealth of further information, including related conferences and newsletters, and other Web resources. We can only hope that this wonderful page will remain up-to-date.

Overall Value

As is the case with such books giving a broad picture of the discipline, some issues are not generally agreed upon. Such disagreements are a sign of a healthy discipline and are essential for its progress. For example, Pfleeger uses the terms "verification" and "validation of requirements or design" in a somewhat controversial sense.

On the whole, however, the book does a good job of introducing the concepts of software engineering. Students will benefit greatly from their investment in this textbook, because it contains enough material to serve as a reference after the course is complete. ❖