Suggested reading

A number of excellent books are available on object technology, only a few of which are briefly described here. Most of these books were chosen for their staying power, they continue to be read and to be cited by other authors. Readers of the *IBM Systems Journal* may be interested in these different perspectives on object technology. (Because it is reviewed in the Books section of this issue, *Succeeding with Objects: Decision Frameworks for Project Management*, by Adele Goldberg and Kenneth S. Rubin, is not included here.)

Object-Oriented Technology: A Manager's Guide, David A. Taylor, Addison-Wesley Publishing Co., Reading, MA, 1990. 147 pp. (ISBN 0-201-56358-4). This book was written while Dr. Taylor was Director of Strategic Planning for Servio Corporation, where it was initially published. When demand for the book exceeded Servio's capability to supply it, Addison-Wesley became the publisher. This book's appeal is in its clear explanations, analogies, and drawings. Not only is the text very readable, but the margins contain a summary of each paragraph, making it easy to skim through the book looking for specific topics. A glossary of terms is included. This book is recommended as a first introduction to object technology.

Object-Oriented Software Construction, Bertrand Meyer, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1988. 534 pp. (ISBN 0-13-629049-3). Dr. Meyer is the developer of the object-oriented language Eiffel, used to illustrate the concepts described here. This book remains a classic because of its discussion of the issues and principles of software engineering and their application to the object paradigm. Programmers and designers concerned with the correctness of programs with respect to their specifications will find this book to be of interest.

Designing Object-Oriented Software, Rebecca Wirfs-Brock, Brian Wilkerson, and Lauren Weiner, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1990. 341 pp. (ISBN 0-13-629825-7). This book does not have a subtitle, but it could have been called "Responsibility-Driven Design." The authors recommend that a designer think of objects first in terms of their responsibilities within the software system being developed, rather than in terms of their attributes (data-driven design) or their behavior. Another important part of their approach is to identify the collaborations between objects needed for the objects to carry out their responsibilities. The same approach is used, at a higher level, for responsibilities and collaborations of subsystems. Examples are included and clearly explained.

Object-Oriented Modeling and Design, James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen, 1991. 491 pp. (ISBN 0-13-629841-9). Perhaps the most widely read of the books listed here, this book introduces analysis and design notations and techniques that have been used, modified, and extended by many other authors. The methodology described, called the Object Modeling Technique (OMT), includes three perspectives of a software system: an object model that shows the relationships between objects and their classes, a dynamic model that describes the events and the state changes that they effect in the system, and a functional model that describes the system's behavior. Analysts and designers of object-oriented software should be familiar with the content of this book.

Object-Oriented Software Engineering: A Use Case Driven Approach, Ivar Jacobson, Magnus Christerson, Patrik Jonsson, and Gunnar Overgaard, 1992. 524 pp. (ISBN 0-201-54435-0). Since the mid-1980s, Dr. Jacobson has been an active participant at object technology conferences, presenting some of the ideas elaborated in this book. A *use case* is a sequence of interactions between a user and a software system during the execution of a well-defined part of the user's job. A use case may be described at a very high level or the description may be quite detailed. Accompanying a use case description is an *interaction diagram* showing the objects involved for a particular use of the system. If the use case description is at a high level, the interaction shown may be between subsystems. As with OMT, use cases and interaction diagrams have been adapted in different ways by other methodologists. This book is recommended for readers who want to better understand Dr. Jacobson's important contribution to object-oriented methods.

Object-Oriented Analysis and Design with Applications, Second Edition, Grady Booch, The Benjamin/Cummings Publishing Co., Redwood City, CA, 1994. 589 pp. (ISBN 0-8053-5340-2). In the first edition of this book, Mr. Booch provided a pragmatic set of techniques for object-oriented design. In this second edition, he describes both micro and macro processes for developing object-oriented software, and he incorporates additional techniques published earlier by other methodologists, including Rebecca Wirfs-Brock, James Rumbaugh, and Ivar Jacobson.

(Both Mr. Rumbaugh and Dr. Jacobson are now associated with Mr. Booch at Rational Corporation.) Both editions of this book include clever drawings by cartoonist Tony Hall, illustrating such important concepts as abstraction, encapsulation, hierarchies, strong typing, concurrency, persistence, inheritance, classification, multiple views, and object collaboration. (Look for the "object-oriented" cat.) Because of its breadth and depth, this book is valuable for all designers of object-oriented software. It contains a glossary of terms and an extensive bibliography.

Design Patterns: Elements of Reusable Object-Oriented Software, Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, Addison-Wesley Publishing Co., Reading, MA, 1995. 395 pp. (ISBN 0-201-63361-2). This book has been reviewed in many publications, including this one. It is cited several times in this object technology theme issue, and one of the papers included here discusses its contents. The authors have compiled a catalog of 23 "design patterns," or ways to solve particular design problems with classes and objects. Each pattern is described using a common template, with sections for intent, motivation, applicability, participants, collaborations, consequences, implementation, sample code, known uses, and related patterns. This book is recommended to all designers and implementors of object-oriented software.

Dictionary of Object Technology: The Definitive Desk Reference, Donald G. Firesmith and Edward M. Eykholt, SIGS Books, Inc., New York, 1995. 603 pp. (ISBN 1-884842-09-7). As object technology has evolved, so has the vocabulary used to describe it. For each term listed in this dictionary, the authors not only provide one or more definitions, they also give the main proponents for each definition. Supplementing the main dictionary are appendices, which provide terms and definitions from the point of view of specific organizations (The Object Management Group and the Object Database Management Group) and languages (Ada95, Common Lisp Object System, C++, Eiffel, and Smalltalk). This book is particularly valuable to instructors, authors, editors, and to anyone else concerned with terms and what they mean in the object technology vocabulary.

Marilyn L. Bates
IBM Systems Journal
Yorktown
New York