Books

Big Blue Java—Complete Guide to Programming Java Applications with IBM Tools, Daniel J. Worden, John Wiley & Sons, Inc., New York, 1999. 572 pp. (ISBN 0-471-36343-X).

Big Blue Java is the latest book written by Daniel J. Worden. Mr. Worden is President of WNS, an authorized IBM Business Partner. This book is a good overview of the various Java** application development tools that IBM offers. The book is appropriate for managers who would like a better understanding of the various IBM software tools, the skills required by the staff, and the type of investment in education and hardware that would be required to do e-business development. It is also appropriate for sales representatives who want an overview of the set of IBM products available for that environment. Many universities are starting to incorporate e-business into their curriculum. This book would be a fine reference to include in a class on managing or leading a software project.

The book begins with an introduction to the Java programming language. It discusses the history of the language, its strengths, and its adoption and endorsement by IBM. For managers who may still be deciding whether to adopt the Java language, the author does a convincing job of touting its benefits.

The author briefly mentions how Java is used on pervasive devices and commits a mistake that is a pet peeve of mine. The author references the over-used analogy of Java on a toaster. I have spent considerable time researching Java on embedded devices and have found so many wonderful useful applications of this technology. Toasters do a good job of making toast. Adding Java to a toaster does not solve any business or consumer problem. It simply adds cost to the appliance. The Java programming language is useful on mainframes, midrange machines, personal computers, and small devices, but let us not discuss kitchen appliances.

The first tool described is VisualAge* for Java. This tool provides a Java integrated development environment for editing, compiling, and debugging Java code. It also provides a visual programming tool that generates Java code, a version control management system to facilitate team programming, and tools to generate code to connect to databases such as DB2* (DATABASE 2*) and to transaction systems such as CICS* (Customer Information Control System). The book concentrates on an older version of VisualAge for Java. VisualAge for Java 3.5 is the current version and offers Enterprise JavaBeans** (EJB**) support. Unfortunately, a few important characteristics of this tool were not explained clearly. The version control management system does not operate on a check-in/check-out file methodology. It operates on a repository, as mentioned, but in an environment of parallel development with a very fine sense of granularity. The granularity is based on classes and methods. The references to checking out a file may lead you to believe it is similar to other popular version control systems.

WebSphere* Studio and NetObjects Fusion** are used to create Web pages. Their use and how they work with other tools such as Allaire HomeSite** and Macromedia Dreamweaver** are shown. The book includes many screen captures so that the reader can see what these tools look like.

The author touches on some of the shortcomings of Java applets and does a fine job of explaining the components of server-side Java. The definitions of JavaServer Pages** (JSP**), servlets, and EJBs and explanations of when you would use them are very clear. The WebSphere Application Server is introduced. This is a complex product that has grown over the past year. Obtaining the most recent documentation on it is highly recommended.

From a commerce perspective, Net.Commerce* and Net.Data* are described. Net.Commerce provides functionality in setting up a storefront on the Web. It provides shopping cart services, catalog management, and secure credit card transaction support. I would have preferred to see a discussion on what Java application programming interfaces are provided.

Several chapters discuss a set of IBM Java frameworks called SanFrancisco*. The author acknowledges the steepness of the learning curve for SanFrancisco. I also found it to be overwhelming initially. SanFrancisco is quite large and has a high memory requirement. The set of Java business objects and processes provided in SanFrancisco are listed. If you are considering SanFrancisco, these chapters are worth reading.

If you have found yourself confused by many of the acronyms of legacy IBM systems, this book is for you. IBM's transactional systems, CICS, messaging system, MQSeries*, and systems management with Tivoli are explained. In addition, an explanation of a new technology, Extensible Markup Language (XML), is also provided.

For managers who want a feel for the complexity of skills required, I recommend you read the chapters on creating the sample Web store. The book shows how skills in the Java language, JavaScript, C++, the macro language of Net.Data, DB2/SQL (Structured Query Language), HTML (HyperText Markup Language), and XML are required. This does not include any of the mainframe legacy skills that might be required. Although these tools make it easier to develop an application, the team of developers needs a wide variety of skills to finish the job.

Given the title, it is no surprise that the author portrays IBM in a positive light. However, in my opinion, it is too favorable. The author does not highlight some of the more major shortcomings of the products. Though the author points out small "gotchas" or minor bugs, he is often too favorable toward IBM on the larger architectural issues. For instance, as each product is described, it is obvious that there is redundancy in function in some areas such as development of the user interface of a Web page. However, with all the overlap, no IBM product, nor a combination of all of them, is sufficient to do a professional Web page design. A non-IBM product must be employed to obtain the artwork demanded for a real application.

The author also discusses the problems of integrating these tools. In addition to the overlap in function, it is difficult to bring work that was completed using one tool into another. Given that all of these tools are developed by one company, it should be expected that a smoother integrated scenario would exist. Part of this problem is that the products do not all support the same Java Development Kit** (JDK**) level. If you intend to do any Java development, you must make sure that the tools you choose support compatible JDK levels.

Among all of the Java books available in bookstores today, it is refreshing to see one that describes IBM products. This is a good overview book. Developers who will actually be writing the code using these products will need to supplement their reading with more detailed documentation.

Sherry Shavor
IBM Software Group
Research Triangle Park
North Carolina

The Object-Oriented Development Process, Tom Rowlett, Prentice Hall PTR, Upper Saddle River, NJ, 2001. 421 pp. (ISBN 0-13-030621-5).

Is this the best book ever written about object technology?

Certainly its clarity is matched only by David A. Taylor's first edition of *OOT: A Manager's Guide*, but Rowlett's book is not intended for beginners. Instead it reveals where knowledgeable developers may safely veer from traditional paths.

And it echoes the truth and terminology of Ivar Jacobson's *Object-Oriented Software Engineering: A Use Case Driven Approach*. But whereas Jacobson illustrates with two examples in C++ and Smalltalk, Rowlett devotes half his book to one example, illustrated with the Java** language. Indeed, Rowlett's chapter on planning and writing test cases, with its easily imitated Java examples, is a major contribution to object-oriented technology.

Although its wisdom about software project management is not as comprehensive as Adele Goldberg and Kenneth S. Rubin's *Succeeding With Objects*, Rowlett's mathematical advice and example of a spreadsheet for managers, in an appendix, go deeper.

Finally, Rowlett's practicality strides step for step with IBM's Object-Oriented Technology Center's (OOTC) *Developing Object-Oriented Software*, but it leaves bigger and deeper footprints by explaining the "whys" and "hows" often only referenced by that IBM bible. For example, Rowlett assumes that his audience is familiar with "scenarios" as defined by IBM, that is, exceptions to the normal use case. But then Rowlett shows how decision tables provide the key to finding and analyzing all the conditions and actions of each use case.

So this book certainly is a contender for "the best," and probably will be regarded as the best by many practitioners inside IBM. They will view it as the *Summa Theologica*—a guide for priests—that explains their OOTC bible by relating its steps not to supernatural dogma but to the Aristotelian logic of Ivar Jacobson.

Practitioners outside IBM will cherish it not only for describing "the" proven process in a way they can understand, but also for weaving 37 correctness questions throughout the process, giving them explicit confidence in activities that always seemed right to them.

Bernard A. Rackmales
IBM Global Services Institute
Stamford, Connecticut

**Trademark or registered trademark of Sun Microsystems, Inc.

Note—The books reviewed are those the Editor thinks might be of interest to our readers. The reviews express the opinions of the reviewers.