Editor: Warren Keuffel ■ wkeuffel@computer.org

A Road Map to Java Software Development

Carlos H.C. Duarte

Java Application Frameworks *by Darren Govoni, Wiley Computer Publishing, New York, 1999, 0-471-32920-4, 410 pp., US $44.99*

Application frameworks are collections of related software artifacts that designers and programmers can instantiate, compose, or customize for specific purposes. They are developed to attempt to facilitate reuse by grouping together artifacts that encapsulate either the knowledge concerning an application domain or best-practice implementations. A set of component specifications for file system handling, written in an object-based programming language, defines a framework. *Java Application Frameworks*, a compendium of Java frameworks, related object-oriented notions, and heuristics, provides excellent coverage of this field.

Your Guide to Software Reuse

There is no goal in software engineering more compelling than effectively reusing software artifacts. The object-oriented approach is one of the best attempts to achieve this objective, and Java is currently the most successful programming language supporting this approach.

Nevertheless, software reuse based solely on programming-language constructs has been disappointing, not only because of the exponential complexity increase when the number of artifacts that could be reused grows, but also because theoretical results show that it is impossible to automate the entire reuse process. Some methodological guidance—as Darren Govoni provides in the form of structuring notions and heuristics—is required to increase the process' granularity level, thus making reuse more effective.

Reuse via Frameworks

Java Application Frameworks is well organized, starting with a description of object-oriented software development that focuses on frameworks. Govoni is pragmatic, presenting framework classification, composition, and complexity without relying on a purely definitional style. Along with the set of presented notions, he discusses issues of practical concern, such as whether to develop or buy a framework and the behaviors that emerge in some complex object systems due to the interaction pattern of their defining components.

Govoni then describes specific Java frameworks, gradually shifting from fine grain, low-level structures to higher-level ones. This is unusual in software engineering books, where the standard approach is to study software artifacts from a high-level perspective and proceed by applying some sort of functional decomposition. Govoni thus successfully orders his chapters, recognizing that understanding object-based notions and low-level frameworks is a prerequisite for reading the succeeding chapters.

Object-Based Notions for Everything

The main body of the book clearly explains how Java packages, frameworks, and software architectures support many standard ideas in software development, such as entity collections, patterns, components, the separation of user interfaces from the appli-

0740-7459/00/$10.00 © 2000 IEEE

cation, remote invocation of functionality, and persistence. Govoni addresses these topics, respectively, with the JDK Collections Framework (Chapter 2), the Gang of Four set of patterns (Chapter 3), the JavaBeans model (Chapter 4), the Swing framework supporting the model-view controller paradigm (Chapter 5), and Remote Method Invocation and Java Database Connectivity (Chapter 7). Suggesting we should adopt frameworks as the units of reuse when developing complex software systems, Govoni also presents a clear and uniform view of the built-in Java 2.0 class hierarchy and of other frequently used Java artifacts that third parties supply.

Good-Looking Technicalities

Another strong aspect of this book is its design—the three artists aren't acknowledged in the foreword for nothing. The humorous art makes reading the book—fully fledged with technical content such as code fragments—a lighter task, and the diagrams help readers understand the complex structures the book discusses. In addition, cross references alert readers to more information on the current and related topics, and the index lets readers use the book as a reference. The publisher's Web site (www.wiley.com/compbooks/govoni) also provides information related to the subject.

Falling Off Track

Unfortunately, readers won't make it through the book without experiencing a few difficulties. At times, such as during the informal definition of the framework notion, it is not clear if some of the properties Govoni discusses are inherent (such as customizability) or desirable (such as extensibility). It also would have been helpful to include a brief explanation of the UML notation with the diagrams. Fortunately, the author deals with these difficulties by providing numerous examples and extensively using the notions he introduces throughout the book. Thus, although the meaning of a few of his informal definitions might not be obvious at first, he clarifies them eventually.

In addition to this, the reader could certainly benefit from more connections with software process and architecture notions. This would offer a better understanding of how frameworks relate to these other notions and could potentially increase the granularity level of supported reuse.

Final Destination

Without a doubt, this is a well-written, carefully designed book presenting an alternative view based on frameworks of the Java paradigm. It complements other works that address the Java programming language alone. It will certainly be a valuable acquisition not only for intermediate and experienced Java software developers seeking to understand by example how to better structure and reuse their work but also as a reference text concerning the most widely used Java application frameworks.

Carlos H.C. Duarte is a member of the technical staff at BNDES, the National Bank of Social and Economic Development in Brazil. Contact him at carlos.duarte@computer.org.

An Easy-to-Use Guide to Use Case Driven Software Development

Martin Fogarty

Use Case Driven Object Modeling With UML: A Practical Approach *by Doug Rosenberg with Kendall Scott, Addison-Wesley, Reading, Mass., 1999, 0-201-43289-7, 165 pp., US $31.95.*

Use Case Driven Object Modeling With UML presents a practical guide to using the Unified Modeling Language to capture user requirements and produce the code necessary to fulfill those requirements. Rather than delving into the many nuances of the notation, Doug Rosenberg proposes a simplified bare-bones usage. (Kendall Scott wrote the book based on conversations with and e-mails from Rosenberg. For simplicity, I won't make this distinction in the review). Rosenberg couples this simplified usage with a methodology that emphasizes getting results—rather than a slavish adherence to the letter of the law. In keeping with this, the writing style is breezy and no nonsense, and the book is mercifully slim. All of this, along with the regular use of enticing words such as *practical*, *stream-lined*, and *simplified*, ought to guarantee a readership among those used to the style of some of the weightier tomes on this subject.

Describing the Methodology

Rosenberg begins by giving his credentials and doing a certain amount of judicious plugging for his company, Iconix. For those studying UML and the Rational Unified Process for the first time, note that the ideas Rosenberg presents predate and, to some extent, foreshadow these approaches.

The basics of the methodology presented are

- identify the real-world domain objects in a domain model, advocating grammatical inspection as a technique to arrive at these objects;
- produce, in parallel, a use case model using the evolving domain model as input (Rosenberg identifies real or paper prototyping as a key enabler for this process.);
- use a process called robustness analysis for first-pass identifica-