Book REVIEW

Stephen P. Berczuk

berczuk@corechange.com

Pattern-Oriented Software Architecture: *A System of* Patterns

by Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal

1996, John Wiley & Sons
ISBN: 0471958697, $39.95

PATTERNS ARE A hot topic in the software development community. The publication of *Design Patterns*[1] in 1995 documented many of the techniques that experienced developers use almost daily in software design.

The authors of *Pattern-Oriented Software Architecture: A System of Patterns* take us further, describing software patterns not just at the lower levels of a system, but also at the architectural level. Their book catalogs eight architectural patterns used by and useful to system architects, and eight design patterns, similar in scope to those described in *Design Patterns.*

It also describes a framework for effectively organizing architectural patterns, design patterns, and idioms. This book makes some basic architectural principles accessible to software designers at all levels and forms the basis for a common design vocabulary.

A pattern describes a solution to a problem in a context. Patterns provide a way to impart system-building expertise to others. In this book, Buschmann et al. provide an introduction to patterns for the uninitiated and a wealth of material for those already familiar with the field. Some of the patterns will be familiar to the experienced designer, others will be new and will provide new perspectives on problems. Far from being just a reference book, this book also provides some context for the growing patterns community by describing the history of patterns, the current community, and future directions.

Pattern-Oriented Software Architecture: A System of Patterns divides its patterns into three categories: architectural patterns, design patterns, and idioms.

Architectural Patterns

According to the authors, an *architectural pattern* expresses a fundamental structural organization schema for software systems. An example presented in the book is the well-known MVC (Model-View-Controller) pattern used in interface design to separate presentation from basic function.

Design Patterns

To define a design pattern, the authors define a design pattern, citing Gamma et al.,[1] as "A commonly recurring structure of communicating components that solves a general design problem within a particular context."

The authors do not just add to the collection of patterns in *Design Patterns,* but they build upon them, with patterns such as Command Processor, which builds on the Command Pattern in Design Patterns to provide a mechanism for separating a request for service from its execution.

Idioms

An idiom is a low-level pattern specific to a programming language. The authors present examples of idioms, showing, for example, how the Singleton pattern can be implemented using two different idioms, depending on whether the implementation language is C++ or Smalltalk. *A System of Patterns* is especially valuable because it is conscious of other pattern work.

Many of the architectural and design patterns in this book build on or are compared to the work presented in *Design Patterns* and other sources, such as the patterns in *Pattern Languages of Program Design*.[2,3]

The authors are fully aware of the context provided by others. Adding to the sense of connectedness, the patterns in the book also cross-reference and are compared to each other, providing insight into when a given pattern is appropriate. The cross referencing also provides a mechanism for avoiding long-winded descriptions of concepts. Instead of explaining how you ensure only master components in the Master–Slave pattern (which involves fault tolerant systems), the authors suggest applying the Singleton pattern from *Design Patterns.*

Summary

It is quite helpful that the patterns in this book have descriptive names, as this simplifies referring to the patterns in design discussions. Once familiar with a pattern, one can simply say, "We need a Broker (or a Proxy) here," providing a detailed bit of information to anyone who has the specific pattern, and giving at least some idea of its meaning to those who don't.

The presentation of the patterns is very effective. Each has a running example, and the roles of the objects in the patterns are diagrammed with CRC (Class-Responsibility-Collaboration) diagrams. Dynamic interactions are illustrated using interaction diagrams, and class structure using the OMT notation.

While patterns are not meant to serve as cookbook recipes, the elements are described clearly enough that you can easily implement and recognize the patterns in your system. The domains of the patterns range from Networking (Forwarder-Receiver), Operating Systems (Micro-Kernel), interactive systems (MVC) and Artificial Intelligence (Blackboard). There are examples in Java and C++.

The authors extensively cite other relevant works, and each pattern has a "Known Uses" section that describes systems in which these patterns have been found, demonstrating that these patterns are based on real experience and that they do have an element of universality.

Far from simply being a catalog of useful patterns a software designer can use as a sourcebook when building complex systems, *A System of Patterns* also explains why patterns helps to simplify the process of building complex architectures. These patterns have the potential to contribute more to the generation of architectures than do those in *Design Patterns,* because they start with high-level problem statements and incorporate other patterns.

Much (but not all) pattern work to date lacks this aspect of building on already discovered patterns to build larger ones, but this is at the heart of the work in architecture[4] that inspired software patterns.

There is a discussion of the patterns community and of future directions. When patterns cross reference each other, the authors adopt the style of *Design Patterns,* which refers to the patterns by the number of the page on which they appear, making them quite easy to find.

At the heart of the work in architecture that inspired software patterns is the practice of building on discovered patterns to produce larger ones

Other aspects of the book are a discussion of the patterns community and of future directions, a good bibliography, and an index of patterns showing not only where the patterns are described but where they are referenced.

Pattern-Oriented Software Architecture: A System of Patterns provides you not only with a collection of patterns that will help you build better systems, but also with an understanding of how to apply the patterns effectively. This book will surely become an important reference for those designing systems and is an excellent companion to *Design Patterns.*

References

1. Gamma, E., et al., *Design Patterns: Elements of Reusable Object-Oriented Software,* 1995, Reading, MA: Addison-Wesley.
2. Coplien, J.O. and D. Schmidt, Eds., *Pattern Languages of Program Design,* 1995, Reading, MA: Addison Wesley.
3. Vlissides, J., J. Coplien, and N. Kerth, Eds., *Pattern Languages of Program Design 2,* 1996, Reading, MA: Addison Wesley
4. Alexander, C., S. Ishikawa, and M. Silverstein, *A Pattern Language,* 1977, Oxford University Press.

Stephen P. Berczuk is with Corechange, LC in Boston, Massachusetts.