Design Patterns
Elements of Reusable Object-Oriented Software

Erich Gamma

Richard Helm

Ralph Johnson

John Vlissides

ADDISON–WESLEY

Boston • San Francisco • New York • Toronto • Montreal
London • Munich • Paris • Madrid
Capetown • Sidney • Tokyo • Singapore • Mexico City

Contents

Preface

This book isn't an introduction to object-oriented technology or design. Many books already do a good job of that. This book assumes you are reasonably proficient in at least one object-oriented programming language, and you should have some experience in object-oriented design as well. You definitely shouldn't have to rush to the nearest dictionary the moment we mention "types" and "polymorphism," or "interface" as opposed to "implementation" inheritance.

On the other hand, this isn't an advanced technical treatise either. It's a book of **design patterns** that describes simple and elegant solutions to specific problems in object-oriented software design. Design patterns capture solutions that have developed and evolved over time. Hence they aren't the designs people tend to generate initially. They reflect untold redesign and recoding as developers have struggled for greater reuse and flexibility in their software. Design patterns capture these solutions in a succinct and easily applied form.

The design patterns require neither unusual language features nor amazing programming tricks with which to astound your friends and managers. All can be implemented in standard object-oriented languages, though they might take a little more work than *ad hoc* solutions. But the extra effort invariably pays dividends in increased flexibility and reusability.

Once you understand the design patterns and have had an "Aha!" (and not just a "Huh?") experience with them, you won't ever think about object-oriented design in the same way. You'll have insights that can make your own designs more flexible, modular, reusable, and understandable—which is why you're interested in object-oriented technology in the first place, right?

A word of warning and encouragement: Don't worry if you don't understand this book completely on the first reading. We didn't understand it all on the first writing! Remember that this isn't a book to read once and put on a shelf. We hope you'll find yourself referring to it again and again for design insights and for inspiration.

This book has had a long gestation. It has seen four countries, three of its authors' marriages, and the birth of two (unrelated) offspring. Many people have had a part in its development. Special thanks are due Bruce Anderson, Kent Beck, and André Weinand for their inspiration and advice. We also thank those who reviewed drafts

of the manuscript: Roger Bielefeld, Grady Booch, Tom Cargill, Marshall Cline, Ralph Hyre, Brian Kernighan, Thomas Laliberty, Mark Lorenz, Arthur Riel, Doug Schmidt, Clovis Tondo, Steve Vinoski, and Rebecca Wirfs-Brock. We are also grateful to the team at Addison-Wesley for their help and patience: Kate Habib, Tiffany Moore, Lisa Raffaele, Pradeepa Siva, and John Wait. Special thanks to Carl Kessler, Danny Sabbah, and Mark Wegman at IBM Research for their unflagging support of this work.

Last but certainly not least, we thank everyone on the Internet and points beyond who commented on versions of the patterns, offered encouraging words, and told us that what we were doing was worthwhile. These people include but are not limited to Jon Avotins, Steve Berczuk, Julian Berdych, Matthias Bohlen, John Brant, Allan Clarke, Paul Chisholm, Jens Coldewey, Dave Collins, Jim Coplien, Don Dwiggins, Gabriele Elia, Doug Felt, Brian Foote, Denis Fortin, Ward Harold, Hermann Hueni, Nayeem Islam, Bikramjit Kalra, Paul Keefer, Thomas Kofler, Doug Lea, Dan LaLiberte, James Long, Ann Louise Luu, Pundi Madhavan, Brian Marick, Robert Martin, Dave McComb, Carl McConnell, Christine Mingins, Hanspeter Mössenböck, Eric Newton, Marianne Ozkan, Roxsan Payette, Larry Podmolik, George Radin, Sita Ramakrishnan, Russ Ramirez, Alexander Ran, Dirk Riehle, Bryan Rosenburg, Aamod Sane, Duri Schmidt, Robert Seidl, Xin Shu, and Bill Walker.

We don't consider this collection of design patterns complete and static; it's more a recording of our current thoughts on design. We welcome comments on it, whether criticisms of our examples, references and known uses we've missed, or design patterns we should have included. You can write us care of Addison-Wesley, or send electronic mail to `design-patterns@cs.uiuc.edu`. You can also obtain softcopy for the code in the Sample Code sections by sending the message "send design pattern source" to `design-patterns-source@cs.uiuc.edu`. And now there's a Web page at `http://st-www.cs.uiuc.edu/users/patterns/DPBook/DPBook.html` for late-breaking information and updates.

Mountain View, California E.G.

Montreal, Quebec R.H.

Urbana, Illinois R.J.

Hawthorne, New York J.V.

August 1994

Foreword

All well-structured object-oriented architectures are full of patterns. Indeed, one of the ways that I measure the quality of an object-oriented system is to judge whether or not its developers have paid careful attention to the common collaborations among its objects. Focusing on such mechanisms during a system's development can yield an architecture that is smaller, simpler, and far more understandable than if these patterns are ignored.

The importance of patterns in crafting complex systems has been long recognized in other disciplines. In particular, Christopher Alexander and his colleagues were perhaps the first to propose the idea of using a pattern language to architect buildings and cities. His ideas and the contributions of others have now taken root in the object-oriented software community. In short, the concept of the design pattern in software provides a key to helping developers leverage the expertise of other skilled architects.

In this book, Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides introduce the principles of design patterns and then offer a catalog of such patterns. Thus, this book makes two important contributions. First, it shows the role that patterns can play in architecting complex systems. Second, it provides a very pragmatic reference to a set of well-engineered patterns that the practicing developer can apply to crafting his or her own specific applications.

I'm honored to have had the opportunity to work directly with some of the authors of this book in architectural design efforts. I have learned much from them, and I suspect that in reading this book, you will also.

Grady Booch

Chief Scientist, Rational Software Corporation

Guide to Readers

This book has two main parts. The first part (Chapters 1 and 2) describes what design patterns are and how they help you design object-oriented software. It includes a design case study that demonstrates how design patterns apply in practice. The second part of the book (Chapters 3, 4, and 5) is a catalog of the actual design patterns.

The catalog makes up the majority of the book. Its chapters divide the design patterns into three types: creational, structural, and behavioral. You can use the catalog in several ways. You can read the catalog from start to finish, or you can just browse from pattern to pattern. Another approach is to study one of the chapters. That will help you see how closely related patterns distinguish themselves.

You can use the references between the patterns as a logical route through the catalog. This approach will give you insight into how patterns relate to each other, how they can be combined with other patterns, and which patterns work well together. Figure 1.1 (page 12) depicts these references graphically.

Yet another way to read the catalog is to use a more problem-directed approach. Skip to Section 1.6 (page 24) to read about some common problems in designing reusable object-oriented software; then read the patterns that address these problems. Some people read the catalog through first and *then* use a problem-directed approach to apply the patterns to their projects.

If you aren't an experienced object-oriented designer, then start with the simplest and most common patterns:

- Abstract Factory (page 87)
- Adapter (139)
- Composite (163)
- Decorator (175)
- Factory Method (107)
- Observer (293)
- Strategy (315)
- Template Method (325)

It's hard to find an object-oriented system that doesn't use at least a couple of these patterns, and large systems use nearly all of them. This subset will help you understand design patterns in particular and good object-oriented design in general.