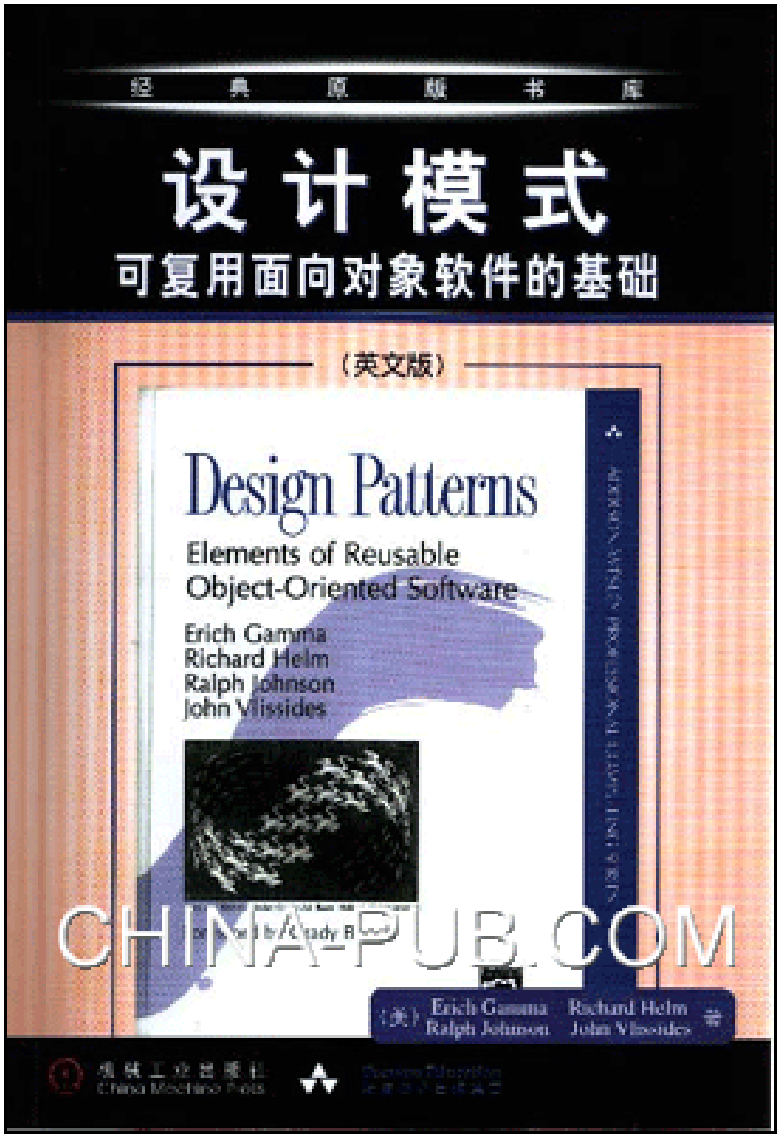

Design Patterns

Elements of Reusable Object-Oriented Software

Produced by KevinZhang

Contents

Preface to CD

As we were writing *Design Patterns*, we knew the patterns we weredescribing had value because they had proven themselves in manydifferent contexts. Our hope was that other software engineers wouldbenefit from these patterns as much as we had.

Now, three years after its debut, we find ourselves both grateful andthrilled by how the book has been received. Lots of people use it.Many tell us the patterns have helped them design and build bettersystems. Many others have been inspired to write their own patterns,and the pool of patterns is growing. And many have commented on whatmight be improved about the book and what they would like to see inthe future.

A recurring comment in all the feedback has been how well-suited thebook is to hypertext. There are numerous cross-references, andchasing references is something a computer can do very well. Sincemuch of the software development process takes place on computers, itwould be natural to have a book like ours as an on-line resource.Observations like these got us excited about the potential of thismedium. So when Mike Hendrickson approached us about turning the bookinto a CD-ROM, we jumped at the chance.

Two years and several megabytes of e-mail later, we're delighted thatyou can finally obtain this edition, the *Design Patterns CD*,and put its unique capabilities to work. Now you can access a patternfrom your computer even when someone has borrowed your book. You can search the text for key words and phrases. It's also considerably easier to incorporate parts of it in your own on-line documentation.And if you travel with a notebook computer, you can keep the bookhandy without lugging an extra two pounds of paper.

Hypertext is a relatively new publishing venue, one we arelearning to use just like everyone else. If you have ideas on howto improve this edition, please send them todesign-patterns-cd@cs.uiuc.edu.If you have questions or suggestions concerning the patternsthemselves, send them to thegang-of-4-patterns@cs.uiuc.edumailing list. (To subscribe, send e-mail to gang-of-4-patterns@cs.uiuc.eduwith the subject "subscribe".) This list has quite a few readers, and many of them can answer questions as well as we can—andusually a lot faster! Also, be sure to check out thePatterns Home Page athttp://hillside.net/patterns/.There you'll find other books and mailing lists on patterns, notto mention conference information and patterns published on-line.

This CD entailed considerable design and implementation work. We areindebted to Mike Hendrickson and the team at Addison-Wesley for theiron-going encouragement and support. Jeff Helgesen, Jason Jones, andDaniel Savarese garner many thanks

for their development effort andfor patience despite what must appear to have been our insatiableappetite for revision. A special acknowledgment is due IBM Research,which continues to underwrite much of this activity. We also thankthe reviewers, including Robert Brunner, Sandeep Dani, Bob Koss, ScottMeyers, Stefan Schulz, and the Patterns Discussion Group at theUniversity of Illinois Urbana-Champaign. Their advice led to at leastone major redesign and several minor ones.

Finally, we thank all who have taken time to comment on *DesignPatterns*. Your feedback has been invaluable to us as we striveto better our understanding and presentation of this material.

Zurich, Switzerland E.G.

Sydney, Australia R.H.

Urbana, Illinois R.J.

Hawthorne, New York J.V.

August 1997

Preface to Book

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 overtime. 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, andRebecca 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 IBMResearch for their unflagging support of this work.

Last but certainly not least, we thank everyone on the Internet andpoints beyond who commented on versions of the patterns, offeredencouraging words, and told us that what we were doing was worthwhile.These people include but are not limited toJon 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 andstatic; it's more a recording of our current thoughts on design. Wewelcome comments on it, whether criticisms of our examples, referencesand known uses we've missed, or design patterns we should haveincluded. You can write us care of Addison-Wesley, or send electronicmail to design-patterns@cs.uiuc.edu. You can also obtainsoftcopy for the code in the Sample Code sections by sending themessage "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

Consider the work of a future software archeologist, tracingthe history of computing. The fossil record will likely show clearstrata: here is a layer formed of assembly language artifacts,there is a layer populated with the skeletons of high orderprogramming languages (with certain calcified legacy partsprobably still showing some signs of life). Each such layer willbe intersected with the imprint of other factors that have shapedthe software landscape: components, residue from the greatoperating system and browser wars, methods, processes, tools. Eachline in this strata marks a definitive event: below that line,computing was this way; above that line, the art of computing hadchanged.

Design Patterns draws such a line of demarcation;this is a work that represents a change in the practice ofcomputing. Erich, Richard, Ralph, and John present a compellingcase for the importance of patterns in crafting complex systems.Additionally, they give us a language of common patterns that canbe used in a variety of domains.

The impact of this work cannot be overstated. As I travel aboutthe world working with projects of varying domains andcomplexities, it is uncommon for me to encounter developers whohave not at least heard of the patterns movement. In the moresuccessful projects, it is quite common to see many of thesedesign patterns actually used.

With this book, the Gang of Four have made a seminalcontribution to software engineering. There is much to learnedfrom them, and much to be actively applied.

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 designobject-oriented software. It includes a design case study thatdemonstrates how design patterns apply in practice. The second partof the book (Chapters 3, 4, and 5) is a catalog of the actual designpatterns.

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

You can use the references between the patterns as a logicalroute through the catalog. This approach will give you insightinto how patterns relate to each other, how they can be combinedwith other patterns, and which patterns work well together. Figure 1.1(page 23) depicts these references graphically.

Yet another way to read the catalog is to use a more problem-directedapproach. Skip to Section 1.6 (page 23) to read about some common problems in designing reusable object-orientedsoftware; then read the patterns that address these problems. Somepeople read the catalog through first and *then* use aproblem-directed approach to apply the patterns to their projects.

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

- Abstract Factory (page 99)
- Adapter (157)
- Composite (183)
- Decorator (196)
- Factory Method (121)
- Observer (326)
- Strategy (349)
- Template Method (360)

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