软件工程系列

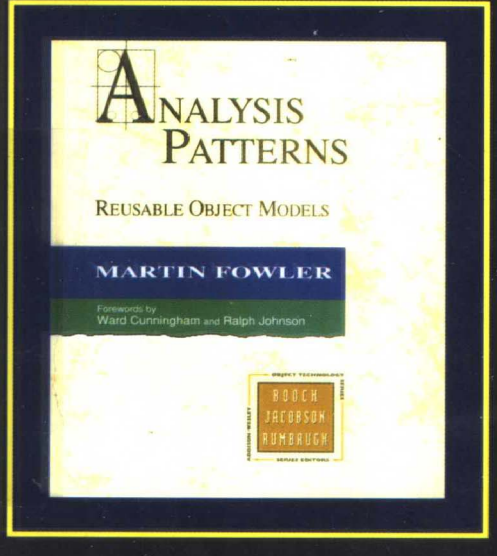
Analysis Patterns
Reusable Object Models

分析模式——可复用对象模型

（影印版）

《设计模式》一书的OOA版

[美] Martin Fowler 著



《重构》和《UML Distilled》作者又一巨著

软件开发和管理人员必读经典

原汁原味，零距离领悟大师思想精髓

中国电力出
www.infopower.c

原版风暴·软件工程系列

Analysis Patterns
Reusable Object Models

分析模式——
可复用对象模型
（影印版）

[美] Martin Fowler 著

中国电力出版社

Analysis Patterns: Reusable Object Models(ISBN 0-201-89542-0)
Martin Fowler

图书在版编目（CIP）数据

分析模式----可复用对象模型 /（美）福勒（Fowler, M.） 著. —影印本. —北京：中国电力出版社，2003
（原版风暴 • 软件工程系列）
ISBN 7-5083-1518-9
Ⅰ.分... Ⅱ.福... Ⅲ.面向对象语言-程序设计-英文 Ⅳ.TP312
中国版本图书馆 CIP 数据核字（2003）第 040107 号

责任编辑：关敏
丛 书 名：原版风暴 • 软件工程系列
书　　名：分析模式——可复用对象模型（影印版）
编　　著：（美）Martin Fowler
出 版 者：中国电力出版社
地址：北京市三里河路6号 邮政编码：100044
电话：（010）88515918 传真：（010）88518169
印　　刷：北京地矿印刷厂
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16　　**印　　张：** 24.25
书　　号：ISBN 7-5083-1518-9
版　　次：2003年7月北京第一版
印　　次：2003年7月第一次印刷
定　　价：48.00 元

Foreword

When the "Gang of Four" was writing *Design Patterns*, we knew that there were lots of software patterns other than object-oriented design patterns. By the time we were through with the book, we had seen distributed programming patterns, user interface patterns, and even patterns of organizing software development groups. However, we hadn't seen any patterns that were clearly object-oriented analysis patterns. Peter Coad's patterns were the closest, but they were a lot like our patterns and it seemed to us that pure analysis patterns should differ more.

I found what I was looking for when I read a draft of Martin Fowler's book, *Analysis Patterns*. Its patterns contain a lot of domain knowledge yet can be used in all kinds of business software. Like the design patterns, they are abstract enough to help your software ride over the bumps of requirement changes but concrete enough to be understandable. They are not the most obvious solutions to modeling problems, yet they rang true to me. I had seen many of these solutions before, and they had worked.

I'm a designer more than a modeler, and I don't have a lot of experience in most of the domains that Martin Fowler describes. Though I felt the patterns were good, I couldn't have a lot of confidence in my feelings. Since I read the book, I have been trying out the patterns on projects and using them in teaching. They work! My confidence grew further when I ran across David Hay's book, *Data Model Patterns*, and realized that, despite their different backgrounds and vocabularies, they saw many of the same patterns. Patterns are supposed to describe reality, not invent a new one, and Martin Fowler accurately described the patterns in object-oriented models of business software. You can have confidence in the patterns he described.

This is not a book of principles that you must learn to apply before they can help you, though Martin describes many modeling principles. It is not a book that you have to read through and practice before it can do you any good. It is a book full of practical patterns that you can use right away. Look for the chapters that match the kind of problem you are working on now, and you will find lots of ideas that will help you. You can read the book chapter by chapter, and each chapter will give you new ideas.

To make the most of this book, you need to know two things. First, many of the patterns are more powerful than they might appear at first. Patterns like Accountability can be applied in nearly any project. Don't read only the chapters that obviously apply to your project, but learn as many patterns as you

can, and try them out to see whether they apply. Second, make sure your coworkers read the book. One of the biggest advantages of patterns is that they help us communicate better. You will find that your team meetings will run more smoothly when you have a common vocabulary. This book will make documentation more consistent and easier to understand. Plus, it will make your coworkers better analysts, and it is more fun to work with people who do a good job!

— *Ralph Johnson*

Foreword

When I look at a software development project, I look for experience. Does the development team have experience doing relevant work? Can they apply their experience to the objects they build? Unfortunately, the answer to these questions is often no.

A growing number of us in the object-oriented development community feel we have misplaced our collective attention for some time. We no longer need to focus on tools, techniques, notations or even code. We already have in our hands the machinery to build great programs. When we fail, we fail because we lack experience.

Martin Fowler has found a way to give us what we need: experience in book form.

He has done for domain objects what Eric Gamma et al. did for implementation objects in their landmark work *Design Patterns: Elements of Reusable Object-Oriented Software*. Martin uses the familiar terminology of our nascent community but in a different way. He uses the word *pattern*, for example, not because he's duplicating or extending Gamma's book (or any of the other new titles bursting onto the market). He calls his written form of experience patterns simply because that is what they are. In his work as a consultant in object modeling information systems, he repeatedly found solutions to recurring problems, and discovered the pattern form in the process.

Martin Fowler easily could have written a book on object-oriented analysis. Luckily, he didn't. Instead we have a book cataloging the result of analysis. Each chapter reports the conclusion of his (and his colleagues') analytic efforts applied to common business problems. The domains addressed vary from medical record keeping to financial derivative trading, with several stops in between. Which chapters apply to you? Amazingly, they all do. Martin places each problem in a context and then offers a solution for that context. You will see familiar aspects in every context. You will recognize the problems. You will appreciate the results. And there it is: experience.

Finally, Martin writes in a personal style, relaying his thoughts and judgments. We feel his respect for his clients and colleagues from whom, he admits, most insights arise. We watch him keep his distance from the vagaries of implementation while still preserving implementability—a tightrope walk that defies direct explanation. As we see into the mind of an expert analyst, we gain a lesson in the how-to of analysis that adds to our own store of experience.

—Ward Cunningham
Cunningham & Cunningham, Inc.

Preface

Not long ago, no books were available on object-oriented analysis and design. Now there are so many that it is impossible for any practitioner to keep up with them all. Most of these books concentrate on teaching a notation, suggesting a simple process for modeling, and illustrating it with a few simple examples. *Analysis Patterns: Reusable Object Models* is a different kind of book. Instead of focusing on the process—how to do modeling—it concentrates on the result of the process—the models themselves.

I am a consultant in object modeling for information systems. Clients ask me to train staff on modeling and to provide mentoring on projects. Much of my skill comes from a knowledge of modeling techniques and how to use them. More important, however, is my experience in actually creating many models and regularly seeing problems repeat themselves. Frequently I find that many aspects of a project revisit problems I have faced before. That experience allows me to reuse models I have built before, improve them, and adapt them to new demands.

Over the last few years, more and more people have also become aware of this phenomenon. We have realized that the typical methodology books, though valuable, only present the first step in a learning process that must also capture the actual things that are built. This realization has flowered into the *patterns* movement. This is a varied group of people, representing many different interests and opinions yet sharing the goal of propagating useful patterns of software systems.

As a result of the diversity of this patterns community, we have had difficulty in defining the term *pattern*. We all think we can recognize a pattern when we see it, we think most of us would agree in most cases, but we cannot come up with a single definition. Here is my definition: A *pattern is an idea that has been useful in one practical context and will probably be useful in others.*

I like to leave the definition quite loose because I wish to stay as close to the underlying motivation of patterns, without adding too many restrictive amendments. A pattern can have many forms, and each form adds specializations that are useful for that kind of pattern. (Section 1.2 discusses the current state of the patterns world and where this book fits in.)

This book is about patterns in analysis, patterns that reflect conceptual structures of business processes rather than actual software implementations. Most of the chapters discuss patterns for various business domains. Such

patterns are hard to classify into traditional vertical areas (manufacturing, finance, health care, and so on) because they are often useful in several areas. These patterns are important because they help us to understand how people perceive the world. It is valuable to base a computer system's design on this perception and, indeed, to change that perception—which is where business process reengineering (BPR) comes in.

Conceptual patterns cannot exist in isolation, however. Conceptual models are only useful to software engineers if they can see how to implement them. In this book I present patterns that can be used to turn conceptual models into software, and I discuss how that software fits into an architecture for a large information system. I also discuss specific implementation tips with the patterns.

I wrote this book because this was the book that I wanted to read when I started out. Modelers will find ideas in this book to help them begin working in a new domain. The patterns contain useful models, the reasoning behind their designs, and when they should and should not be applied. With this information a modeler can adapt the models to fit a specific problem.

The patterns in this book can also be used in reviewing models—to see what might have been left out and to suggest some alternatives that may lead to improvement. When I review a project, I usually compare what I see with the patterns I have learned from previous work. I have found that being aware of patterns in my work helps me to apply my past experiences more easily. Patterns like this also uncover modeling issues that go beyond what can be covered in a simple text book. By discussing why we model things the way we do, we gain a greater understanding of how to improve our modeling, even if we don't use the patterns directly.

Structure of this Book

This book is divided into two sections. The first section covers analysis patterns, which are patterns from conceptual business models. They provide key abstractions from domains such as trading, measurement, accounting, and organizational relationships. The patterns are conceptual because they represent the way people think about the business, rather than the way a computer system is designed. The chapters in this section stress alternative patterns that can be used, and the strengths and weaknesses of those alternatives. Although each pattern will clearly be useful to those working in the same domain, the basic pattern is often useful in other domains.

The second section focuses on support patterns, which help you use analysis patterns. Support patterns show how analysis patterns fit into an information systems architecture, how the constructs of conceptual models

turn into software interfaces and implementations, and how certain advanced modeling constructs relate to simpler structures.

To describe these patterns, I need a notation. The appendix provides a brief discussion of the notation I use and what the symbols mean. I do not use a single method but prefer to mix techniques from different methods. The appendix is not designed to be a tutorial on techniques, but it should provide an outline and refresh your memory. It also tells you where to find a tutorial on the techniques I use.

Each section is divided into chapters. Each chapter on analysis patterns contains patterns that are related by a loose notion of subject area, influenced by the projects that spawned them. This organization reflects the fact that any pattern must come from a practical context. Each pattern appears in its own subsection within a chapter. I do not use any of the formal headings for patterns that are used by some patterns authors (see Section 1.2.2). I describe each pattern in a form that is as close to the original project form as is reasonable, with a minimum of abstraction. I add examples to show the use of the pattern within its original domain and also to suggest how the pattern might be used in other domains. One of the greatest difficulties of patterns is abstracting them into other domains; I follow the principle that this should be left to the reader (see Section 1.2.3).

This book is thus a catalog, rather than a book to be read from cover to cover. I have tried to write each chapter in such a way that it can be read independently from the other chapters. (This is not always possible, however. Whenever a chapter requires that another chapter be read first, I say so in the chapter introduction.) Each chapter has an introduction that explains the general subject area of the chapter, summarizes the patterns in the chapter, and says what projects the patterns originated from.

How to Read this Book

I suggest reading all of Chapter 1 first and then reading each chapter introduction. Then feel free to delve into the chapters in any order you like. If you are not familiar with the approach I take to modeling, or the notation and concepts I use, read the appendix. The Table of Patterns gives a brief summary of what each pattern is about, so you can use that to help you explore or to find a pattern when you come back to the book at a later time. It is important to stress that each pattern in this book is useful outside the domain that gave it birth. Thus I encourage you to look into chapters that you might think are outside your field of interest. For example, I found that models of observation and measurement designed for health care proved to be very useful for corporate financial analysis.

Who Should Read this Book

This book can be useful to a range of readers, although different readers will learn different things from it and may need some different preparations.

I expect my biggest audience to be *analysts and designers* of object-oriented (OO) computer systems, particularly those working at the analysis end. Such readers should have made at least some use of an OO analysis and design method. This book does not provide any introduction to this subject, so I would suggest first reading a book on OO analysis and design if you are new to this field. I must stress that the patterns in this book are conceptual in nature, and I use a very conceptual approach to modeling. This leads to some stylistic differences from those texts that use a more implementation-based approach to modeling.

A small, but very important, audience consists of those people who act as *domain experts for a modeling project.* Such readers do not require a knowledge of computers but do need to know about conceptual modeling. One of the main reasons I use conceptual models in this book is to make things easier for this group of readers. The modeling project here may be analysis for computer system development or BPR. I have taught many professionals (including doctors, financial traders, accountants, nurses, and payroll supervisors) this kind of modeling and have found that a software background is neither an advantage nor a disadvantage to conceptual modeling. The business model patterns are as much about business modeling as they are about computer systems analysis (see Section 1.4). Any such reader should take a course on OO analysis that stresses the conceptual aspect. (Odell's book [1] is particularly valuable in this respect.)

I hope many *programmers* will delve between these covers, although some programmers may take exception to the lack of code and the conceptual slant. For these readers I suggest you take particular note of Chapter 14, which should help to explain the relationship between the conceptual models and the resulting software.

This is an object-oriented book, and I do not hesitate in proclaiming my belief that the object-oriented approach is the superior way to develop software. These models, however, are primarily conceptual models, and many *data modelers* have had a long tradition of using conceptual (or logical) models. Data modelers should find many of the patterns useful, particularly if they use more advanced semantic techniques. The object-oriented features of the models will reveal many of the differences between object-oriented and traditional approaches. I would encourage such readers to use this book in conjunction with an OO analysis book that stresses the conceptual side of modeling and the links between OO and semantic data modeling.

Managers will find the book useful as a starting point for development activity. Starting from a pattern can help to clarify goals, and project planning can take advantage of the broad ground that patterns map out.

I have not aimed this book at *students*. I've written it more for the professional software engineer. I hope, however, that some students will take a look. When I was learning analysis and design, I found it difficult because there were few good examples I could learn from, examples that came out of the world outside the university. Just as looking at good code can teach you a lot about programming, looking at good models can teach you a lot about analysis and design.

A Living Book

Every author I know shares a frustration: Once a book is published it is fixed. The book spreads its advice around the community, yet the author has little way of expressing changes. I know how much I keep learning, and I am sure this learning will modify my ideas. I want these changes to be passed on to my readers.

With this book, Addison-Wesley will provide a web site <http://www.aw.com/cp/fowler.html> which will be used to pass on further materials to keep this book alive. At this stage I am not sure exactly what it will contain, but I expect the following:

- any new things I learn about the patterns in the book.
- answers to questions about the book
- useful commentary from others about the patterns
- new analysis patterns by myself, and by others
- when the Unified Modeling Notation appears (or whatever it is called by then) I will redraw all the diagrams in the book in the new notation and put them on the site.

This site will be a complement to the book, so keep an eye on it and use it to let me know how to improve and develop the ideas between these pages.

Acknowledgments

Any author is indebted to many others who help. For this book this is particularly true since so many of the patterns were built with the help of my clients, colleagues, and friends. I would like to give my sincere thanks to the following, both named and implied.

First and foremost, Jim Odell has been an essential part of my career. He has taught me much about developing information systems and has been a

constant source of inspiration, helpful advice, and strange humor. I can safely say that without his support this book would not have happened.

The team at Coopers & Lybrand in London helped with much of the early work and helped pass many evenings at Smithfield's.

John Edwards formed many of my early ideas about conceptual modeling and its role in software development, as well as introducing me to many interesting ideas, including those of Christopher Alexander.

John Hope urged me to think of the domain first and technology second, as well as casting a helpful spell at several key points in my career.

Tom Cairns and Mark Thursz, doctors at St. Mary's Hospital in London, worked with me in developing the health care models that form the basis of Chapters 2, 3, and 8. They are proof that a computer background is not necessary to be a top-class conceptual modeler. Mark also was a willing source for health care examples with impressive-sounding medical terminology.

The health care projects also involved many software and health care professionals from St. Mary's, the Hospital for Sick Children (HSC), St. Thomas's Hospital, and the University of Wales. Anne Casey, a nurse at HSC, and Hazim Timimi, an analyst, helped put together the final Cosmos model. Gerry Gold set up this work and made sure it kept going.

Brad Kain has had a great impact on my thinking on reuse and components, as well as undertaking the important task of showing me the nightlife of Boston.

Applying the health care models to corporate finance in Chapter 4 was the experience that, for me, proved the usefulness of analysis patterns across different domains. Lynne Halpin and Craig Lockwood led the MBFW team at Xerox, and Vivek Salgar got our conceptual ideas into the brutal reality of C++.

David Creager, Steve Shepherd, and their team at Citibank worked with me in developing the models from which I drew the financial patterns in Chapters 9–11. They also further developed many of the architectural ideas of Chapter 12 from their health care origins, and taught me much about the frenetic life in The City.

Fred Peel set up and maintained my work at Citibank, when not scaring me with his driving. Daniel Poon and Hazim Timimi from Valbecc got many of my fuzzy ideas into detailed specifications.

The accounting patterns in Chapter 6 have had a long gestation. Tom Daly, Peter Swettenham, Tom Hadfield, and their respective teams developed models that gave birth to the patterns in this book. Rich Garzaniti got my accounting terminology sorted out. Kent Beck did much to improve my Smalltalk.

Chapter 14 was written with the help of James Odell.

I have been very much a latecomer to the patterns community, getting to know it well only after most of this book was written. It is a very open and friendly group that has done much to encourage my work. Kent Beck, Ward Cunningham, and Jim Coplein encouraged me to get involved with the community and to develop my ideas as patterns. Ralph Johnson provided particularly helpful comments on the first draft of this book.

I have had first-class comments from my many reviewers whom I would like to name: Dave Collins, Ward Cunningham (Cunningham & Cunningham, Inc.), Henry A. Etlinger (Department of Computer Science, RIT), Donald G. Firesmith (Knowledge Systems Corporation), Erich Gamma, Adele Goldberg, Tom Hadfield (TesserAct Technology), Lynne Halpin (Netscape Communications), Brian Henderson-Sellers, Neil Hunt (Pure Software), Ralph E. Johnson (University of Illinois at Urbana-Champaign), Jean-Pierre Kuilboer (University of Massachusetts, Boston), Patrick D. Logan (Intel Corporation), James Odell, Charles Richter (Objective Engineering, Inc.), Douglas C. Schmidt (Washington University), and Dan Tasker. I will mention that Don Firesmith went above the call of duty in tracking down problems that needed to be fixed.

As this is my first book, I'm particularly grateful to those at Addison-Wesley who helped me through the process. Carter Shanklin directed affairs and assembled a formidable panel of reviewers with much assistance from Angela Buenning. Teri Hyde coordinated the book production on a painfully tight schedule and Barbara Conway rescued my prose from its usual erratic state, and ruthlessly eliminated my native accent.

References

1. Martin, J., and J. Odell. *Object-Oriented Methods: A Foundation.* Englewood Cliffs, NJ: Prentice-Hall, 1995.

Contents