HZ BOOKS

PEARSON
Addison Wesley

经 典 原 版 书 库

面向方面的分析与设计
Theme方法

(英文版)

ASPECT-ORIENTED ANALYSIS AND DESIGN
THE THEME APPROACH

SIOBHÁN CLARKE
ELISA BANIASSAD

OBJECT TECHNOLOGY SERIES
ADDISON-WESLEY
BOOCH
JACOBSON
RUMBAUGH
SERIES EDITORS

(爱尔兰) Siobhán Clarke
Elisa Baniassad 著

机械工业出版社
China Machine Press

经　典　原　版　书　库

面向方面的分析与设计

Theme方法

(英文版)

Aspect-Oriented Analysis and Design

The Theme Approach

(爱尔兰) Siobhán Clarke
Elisa Baniassad 著

机械工业出版社
China Machine Press

本书版权登记号：图字：01-2005-5836

图书在版编目（CIP）数据

面向方面的分析与设计：Theme方法（英文版）/（爱尔兰）克拉克（Clarke, S.）等著. －北京：机械工业出版社，2006.1
（经典原版书库）
书名原文：Aspect-Oriented Analysis and Design: The Theme Approach
ISBN 7-111-17975-7

Ⅰ.面…　Ⅱ.克…　Ⅲ.面向对象语言－应用软件－英文　Ⅳ.TP312

中国版本图书馆CIP数据核字（2005）第141698号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）
责任编辑：迟振春
北京诚信伟业印刷有限公司印刷·新华书店北京发行所发行
2006年1月第1版第1次印刷
718mm × 1020mm 1/16 · 24.75印张
印数：0 001-3 000册
定价：49.00元

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭櫫了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短、从业人员较少的现状下，美国等发达国家在其计算机科学发展的几十年间积淀的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章图文信息有限公司较早意识到“出版要为教育服务”。自1998年开始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall，Addison-Wesley，McGraw-Hill，Morgan Kaufmann等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出Tanenbaum，Stroustrup，Kernighan，Jim Gray等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及庋藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都步入一个新的阶段。为此，华章公司将加大引进教材的力度，在“华章教育”的总规划之下出版三个系列的计算机教材：除“计算机科学丛书”之外，对影印版的教材，则单独开辟出“经典原版书库”；同时，引进全美通行的教学辅导书“Schaum's Outlines”系列组成“全美经典学习指导系列”。为了保证这三套丛书的权威性，同时也为了更好地为学校和老师们服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔

滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

这三套丛书是响应教育部提出的使用外版教材的号召，为国内高校的计算机及相关专业的教学度身订造的。其中许多教材均已为M. I. T.，Stanford，U.C. Berkeley，C. M. U. 等世界名牌大学所采用。不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程，而且各具特色——有的出自语言设计者之手、有的历经三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下，读者必将在计算机科学的宫殿中由登堂而入室。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证，但我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。教材的出版只是我们的后续服务的起点。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

电子邮件：hzjsj@hzbook.com
联系电话：(010) 68995264
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

专家指导委员会

（按姓氏笔画顺序）

尤晋元	王　珊	冯博琴	史忠植	史美林
石教英	吕　建	孙玉芳	吴世忠	吴时霖
张立昂	李伟琴	李师贤	李建中	杨冬青
邵维忠	陆丽娜	陆鑫达	陈向群	周伯生
周克定	周傲英	孟小峰	岳丽华	范　明
郑国梁	施伯乐	钟玉琢	唐世渭	袁崇义
高传善	梅　宏	程　旭	程时端	谢希仁
裘宗燕	戴　葵			

For Padraic, with my love and thanks.

—Siobhán

To Ryan.

—Elisa

Preface

Aspects are a natural evolution of the object-oriented paradigm. They provide a solution to some difficulties you may have encountered with modularizing your object-oriented code: sometimes functionality just doesn't fit! You've probably found yourself repeating the same lines of code in lots of different object-oriented classes because those classes each need that functionality, and so you can't easily wrap it up in a single place. Good examples of this kind of code are audit trails, transaction handling, concurrency management, and so on. You can now modularize such code with aspects.

We've seen similar levels of enthusiasm with adopting aspects as there were with adopting objects—an enthusiasm we share; but starting out with aspects can be a tricky business. Making the shift to aspect-oriented thinking may not be as tough as many people found the shift to object-oriented thinking, but aspects still might take a little getting used to. The big question that springs to mind when trying out aspect-orientation for the first time is "What are my aspects?" and early adopters have taken various approaches to try to address it.

We've heard of practitioners trying to apply aspects, but who can't think of any except those typical, and somewhat trivial ones. The usual examples are out there to be tried: logging, debugging, coordination. But to make fluent use of aspects, you also want to be able to use them for concerns that are specific to your own code.

We've heard of others who have made so many tiny aspects that the classes in their core system have no functionality whatsoever! They achieved so much "separation of concerns" that they could hardly work out the control-flow of their programs.

Another typical approach to answering the "what are my aspects" question is to just program vanilla OO code, and then try to spot the functionality that doesn't quite fit in. That approach has some serious disadvantages. In particular, it keeps you from being able to reason about aspects until you start to code. After all, you probably don't wait until you start to write code before figuring out what your classes should be (even if they're only a starting point). It's the same deal with aspects.

Besides being somewhat confusing, early adoption of a paradigm has some risks. Aspect-orientation is in an exciting phase of growth, but that means that new languages and new possibilities are coming out frequently, and that the basic notions of an "aspect" shifts subtly as new philosophies are revealed. There are different styles of decomposition, even within aspect-orientation. Which should you choose?

In this book we describe the Theme approach for identifying aspects in requirements, and modeling them at design. A major strength of the Theme approach is that it allows you to identify and model aspects regardless of the aspect-oriented programming language you choose. Our intention in developing the Theme approach was to enable it to withstand these shifts by keeping it separate from any particular programming language and by offering a general-purpose way to identify and describe aspects, regardless of their definition at the code level.

In addition to talking about the Theme approach and how to apply it, we also describe the different "worlds" of aspect-orientation, and how the Theme approach fits into them. You will come away from reading this book with not just tools for analysis and design, but also with an understanding of the general field of AO as it stands today. That knowledge will help you make more informed choices when picking an aspect-oriented implementation language, and decomposition paradigm.

Audience

For a wide range of situations, AOSD improves the software development process. This book offers a high-level introduction to the aspect-oriented approach, and gives instruction on a useful approach for identifying aspects in requirements, and for designing them in an aspect-oriented way using UML with a small number of extensions.

We have written this book for practitioners and early adopters of aspect-orientation. This book will be particularly helpful for those who are trying to answer the common questions of "What is an aspect?" and "Which aspects should I be coding?" This book gives you a starting point for thinking about aspects, and accounting for them in your requirements and design.

Even if you've been using aspect-oriented languages for a while, you can read this book to learn more about identifying aspect functionality in requirements documentation and how to plan for aspect-design and implementation. The Theme approach gives a flexible way to identify aspect-functionality, and a UML-based design modeling language that can describe aspects independently of programming language. Whatever your aspect-oriented programming language, the analysis and design approach and principles described in this book will be helpful and informative.

Of course, this book would also be helpful to academics or students wishing to learn more about the aspect-oriented paradigm.

For all readers, we assume that you are familiar with the object-oriented paradigm, and are comfortable with the UML notation.

History of Aspect-Oriented Analysis and Design and The Theme Approach

Analysis and design approaches for software engineering paradigms have traditionally emerged after people have explored the ideas at the programming level for a while. From there, application of the ideas tends to move backwards through the software lifecycle. This is true of aspect-oriented analysis and design and so before we look at the origins of Theme, we'll first take a quick look at what was happening at the code level from the early 1990s.

It's hard to choose where to begin a history of aspect-oriented programming, as a lot of the work we talk about as AOP emerged from the creators' previous work in the general area. We could also take a broader view in the larger context of software engineering, as many researchers have been working on improving software modularization for decades in work that is not viewed under the "Aspect" umbrella. We'll take the easy way out here, and simply mention the four main approaches to improved modularization that are popularly regarded as being the origins of aspect-oriented software development.

The most well known approach is the one popularized by the AspectJ language, which was first developed by a team from Xerox PARC in 1997, led by Gregor Kiczales. Previously, the team had worked on metaobject protocols and reflection, with ideas evolving to the modularisation of "crosscutting" concerns. Meanwhile, in 1993, a team from IBM T.J. Watson Research Center, led by William Harrison and Harold Ossher, published work on "subject-oriented programming". Subject-oriented programming (and its later incarnations as multi-dimensional separation of concerns co-led by Peri Tarr) looks at flexible decomposition and composition of software modules based on different dimensions of concern. The academic community was also hard at work; the next two approaches emerged from university research. At the University of Twente in The Netherlands, Mehmet Aksit and his team had been working on Composition Filters since the early 1990s. With this approach, behavior is modularized in "filters" that can be used to capture and enhance the execution of object behavior. Karl Lieberherr at Northeastern University in the US defined the Demeter Method in the mid 1990s that provides abstractions of the class structure and navigation to support better separation of this knowledge from an operation's behavior. Crista Lopes worked with both Karl Liberherr and Gregor Kiczales in developing D-Java, and the first official set of "Aspect languages" in 1997. Fast-forward to 2004 and aspect-oriented programming languages are coming out of the woodwork! Notably, though, each of the new ones is rooted in principles that originated from one or more of the original four.

Back to analysis and design. In those early years of aspect-oriented programming, there was little to no work being published on supporting aspect-like principles at earlier stages in the development lifecycle. The Theme approach to aspect-oriented design was the first approach to incorporate aspects into the UML, with Siobhán giving some early ideas their first "outing" at an OOPSLA workshop in 1997. Its further formulation was worked on in collaboration with IBM Research, in particular with Peri Tarr, Harold Ossher and William Harrison, and also with Robert Walker from (at the time) the University of British Columbia. The design model benefited considerably from subject-oriented programming principles to the extent that it was labeled "subject-oriented design" for a few years. However, as you'll see reading this book, we see the Theme approach as encompassing different aspect schools of thought, and so Siobhán re-labeled the work on "subject-oriented design" to "Theme/UML" in 2001.

Identifying and visualizing concerns in documentation was initially explored by Elisa with Gail Murphy of University of British Columbia, and Christa Schwanninger of Siemens AG. That work motivated Theme/Doc's emergence in 2003 as the aspect-oriented analysis part of the Theme approach. Theme/Doc is intended as a complement to your existing analysis process, and is the missing link between having a set of requirements, and knowing what aspects should be designed using Theme/UML.

In forming the Theme approach, we kept in mind the real goals of the programmer: to understand the problem space (the requirements), and design appropriately. Our goal was to create an approach that allows the developer to map requirements to design to code. Theme/Doc and Theme/UML provide this mechanism. Theme/Doc helps you find the aspects in your requirements. Theme/UML helps you design them. Together, they form the Theme approach.

How to Read This Book

Of course, the most straightforward way to read this book is from start to finish. The book follows the basic structure of introduction and motivation (Chapters 1 and 2), overview and illustration of the approach (Chapters 3-6), guidance on mapping your designs to some AOP languages in Chapter 7 and examples of its application (Chapters 8 and 9).

However, different parts of the book may be of more interest than others, depending on your perspective. If you're not sure what an aspect even is, then Chapters 1, 3 and 4 will be of lots of help. They go over the basic concepts and walk you through finding aspects in a set of requirements.

If you're not convinced aspects are all that great, and are asking the question "Why do we need them anyway?" then Chapter 2 will be for you. Chapters 8 and 9 will also provide you with examples of how aspects can be applied in different kinds of systems.

If you'd like instruction on capturing aspects in design, then Chapters 5 and 6, which provide details of Theme/UML will walk you through designing the aspects and the core of your system, and on capturing the specification of their composition.

Acknowledgments

We very gratefully acknowledge the assistance of many people in the writing of this book.

Several people contributed to the technical content. Mary Lee worked on applying the Theme approach in the early stages. Andrew Jackson contributed most of the code samples in the book. Harold Ossher and William Harrison contributed the CME code. Alan Gray, David McKitterick, Tonya McMorrow, Karl Quinn, and Conor Ryan were on the development team of the original Crystal Game on which the book's example is based. Christa Schwanninger and Ivana Dusparic advised and contributed the case studies. We had interesting discussions with Vinny Cahill on developing peer-to-peer systems.

We also thank those who reviewed copies of the manuscript throughout its creation: Ron Bodkin, Adrian Colver, Yvonne Coady, João M. Fernandes, Charles Haley, Paul Holser, Wes Isberg, Gregor Kiczales, Ramnivas Laddad, Awais Rashid, Ryan van Roode, Michael J. Ward, and Tim Walsh.

Throughout the history of the Theme approach, William Harrison, Stuart Kent, Michael Kircher, Gail Murphy, John Murphy, Harold Ossher, Christa Schwanninger, Peri Tarr, and Robert Walker were invaluable influencers and contributors, for which we are indebted.

Finally, we thank all at Addison-Wesley for making this possible: Kristy Hart, Brenda Mulligan, Mary O'Brien, Kerry Reardon, and Chris Zahn.

Praise for *Aspect-Oriented Analysis and Design*

"Developers who are using aspect-oriented programming will appreciate this contribution to aspect-oriented analysis and design. The authors are pioneers in this area and have elaborated on past research to produce a detailed methodology and notation for early aspects."

—*Ron Bodkin, Chief Technology Officer*
New Aspects of Software

"Aspect-orientation is a powerful approach for programming complex systems. There is a lot to be gained from applying this approach during modeling and designing, as well. The Theme approach in this book represents an important advancement in AOP adoption by providing practitioners means to apply aspect-orientation early on."

—*Ramnivas Laddad*
Author of Aspect J in Action

"Clarke & Baniassad have written an interesting book that shows how to use aspects to solve a difficult problem: composing independent program fragments with overlapping functionality. The included case studies well illustrate the principles. I recommend the book."

—*Charles B. Haley*
Research Fellow, The Open University

"This book presents a very useful set of techniques for helping software developers to identify the aspects. I am sure that this book will rapidly become a landmark reference for the software community!"

—*João M. Fernandes*
Ph.D., Universidade do Minho

About the Authors

Siobhán Clarke

Siobhán Clarke is a lecturer at Trinity College, Dublin. She holds BS (1986) and PhD (2000) degrees from Dublin City University. Siobhán worked for IBM Ireland Ltd. in various leading software engineering roles from 1986 to 1997. In 1997, she started her PhD, which was based on extending the modularization and composition capabilities of UML. This work evolved into Theme/UML.

Siobhán's current research focus is on design and programming models for mobile, context-aware systems. The complexities associated with developing such systems require advanced software engineering techniques. In particular, she is investigating and extending aspect-oriented software development (AOSD) techniques as a means to address these complexities.

Siobhán has served on the program committees of AOSD and UML conferences, and on the organizing committees for AOSD, and MoDELS. She has co-organized and/or been on the program committee for multiple workshops at conferences such as OOPSLA, ECOOP, ICSE, AOSD, and UML in the area of design and programming models and context-aware computing. She is on the editorial boards of IEEE Internet Computing and the Springer Transactions on AOSD.

Elisa Baniassad

Elisa Baniassad is a professor at the Chinese University of Hong Kong. She received her PhD in 2002 from the University of British Columbia, Canada, where she worked with Gail Murphy. Elisa then carried out a postdoctoral fellowship, funded by the National Science and Engineering Council of Canada and held at Trinity College, Dublin.

Elisa first became intrigued by the AO world during a visit to Xerox PARC in 1997 while involved in some of the earliest empirical work on AOP. She then began looking at concerns in documentation with her PhD work on Design Pattern Rationale Graphs: finding concerns in design patterns text and tracing them through design to code. This work included broader research into how programmers relate to both their code and to the documentation upon which they rely. Her main group of victims were gathered by Christa Schwanninger of Siemens AG.

She then turned to investigating how to bridge from requirements to aspect-oriented design and started research on Theme/Doc. That work is currently ongoing, involving empirical studies of programmers and tool development.

Elisa is involved in several software engineering conferences and has served on the organizing and/or program committees of OOPSLA, ECOOP, and AOSD. She has also published papers at these conferences as well as at ICSE. Elisa is an organizer of the Early Aspects workshop that is typically held at AOSD and OOPSLA.