

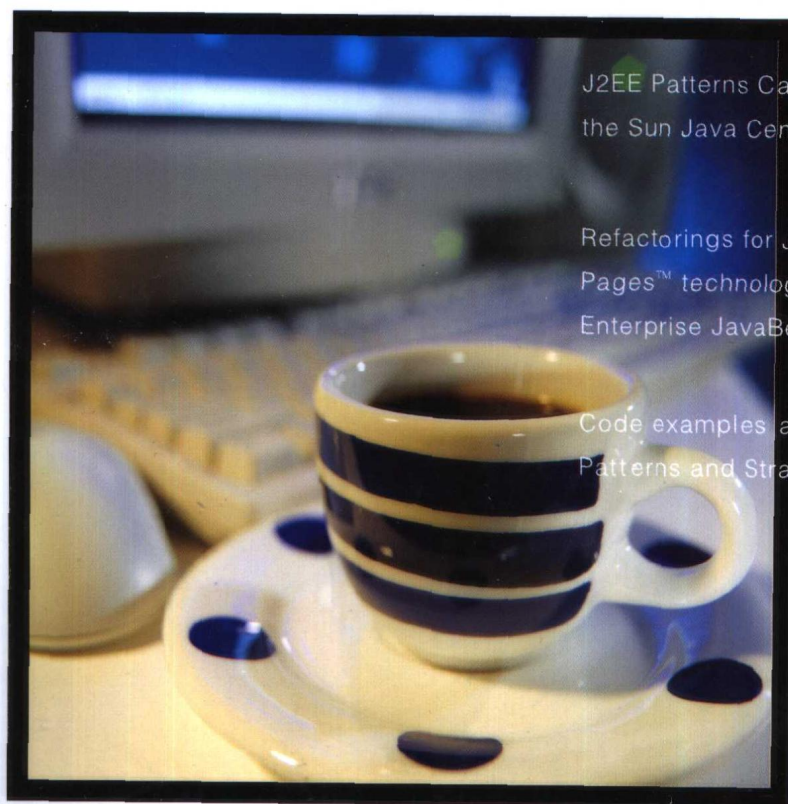
Java程序员书库

# J2EE 核心模式

(影印版)

Core J2EE™ Patterns

Best Practices and Design Strategies



J2EE Patterns Catalog from  
the Sun Java Center

Refactorings for JavaServer  
Pages™ technology, Servlets, and  
Enterprise JavaBeans™ architecture

Code examples and UML diagrams for  
Patterns and Strategies



Deepak Alur  
(美) John Crupi 编著  
Dan Malks



科学出版社

[www.sciencep.com](http://www.sciencep.com)

Java 程序员书库

# J2EE 核 心 模 式

(影印版)

Core J2EE<sup>TM</sup> Patterns  
Best Practices and Design Strategies

Deepak Alur  
(美) John Crupi 编著  
Dan Malks

科 学 出 版 社

北 京

图字: 01-2003-6988 号

## 内 容 简 介

本书主要讲述企业 Java 2 平台(J2EE)关键技术的模式、最佳实践、设计策略和经过验证的解决方案。涉及 J2EE 包括的 15 个模式的分类和大量的策略, 便于读者更好地掌握 Java 技术。

本书适合 J2EE 的爱好者、程序员、设计师、开发者和技术管理者参考。

English reprint copyright©2003 by Science Press and Pearson Education Asia Limited.

Original English language title: Core J2EE™ Patterns: Best Practices and Design Strategies, 1<sup>st</sup> Edition by Deepak Alur, John Crupi and Dan Malks, Copyright©2001

ISBN 0-13-064884-1

All Rights Reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Prentice Hall PTR.

For sale and distribution in the People's Republic of China exclusively (except Taiwan, Hong Kong SAR and Macao SAR).

仅限于中华人民共和国境内(不包括中国香港、澳门特别行政区和中国台湾地区)销售发行。

本书封面贴有 Pearson Education(培生教育出版集团)激光防伪标签。无标签者不得销售。

### 图书在版编目(CIP)数据

J2EE 核心模式=Core J2EE™ Patterns: Best Practices and Design Strategies/ (美) Deepak Alur 等编著. —影印本. —北京: 科学出版社, 2004

(Java 程序员书库)

ISBN 7-03-012465-0

I. J... II. A... III. Java 语言—程序设计—英文 IV. TP312

中国版本图书馆 CIP 数据核字 (2003) 第 099960 号

策划编辑: 李佩乾/责任编辑: 舒 立

责任印制: 吕春珉/封面制作: 北新华文平面设计室

**科学出版社 出版**

北京东黄城根北街16号

邮政编码: 100717

<http://www.sciencep.com>

**双青印刷厂 印刷**

科学出版社发行 各地新华书店经销

\*

2004 年 1 月第 一 版 开本: 787×960 1/16

2004 年 1 月第一次印刷 印张: 30 1/2

印数: 1—3 000 字数: 579 000

定价: 50.00 元

(如有印装质量问题, 我社负责调换〈环伟〉)

# 影印前言

程序设计在于处理复杂性：问题的复杂性和所用的程序设计工具的复杂性。Java 的魅力在于其本身的低复杂性，同时又能很好地处理高度复杂的问题。Java 程序的开发周期只有类似的 C++ 程序的一半甚至更少，而且 Java 可以方便地处理复杂软件问题：多线程、分布式、跨平台、安全性等。Java 从诞生到现在，已经广泛应用于几乎所有类型软件系统的构建。尤其在基于 Web 的系统开发中，Java 技术具有独特的优势。熟悉 Java 历史的人都知道，Java 的前身——编程语言 Oak 就是致力于电子产品互连的语言，Internet 的发展导致了 Oak 的重生和 Java 的广为流行。现在，J2EE 技术已经成为企业级 Web 应用系统的标准平台。

好的程序设计人员不仅仅要掌握优秀的编程工具，更需要掌握优秀的编程思想。随着面向对象编程技术数十年的发展，人们开始提炼和总结面向对象编程中行之有效的、具有一定普遍意义的方法，即面向对象的设计模式。以 Gamma、Helm、Johnson 和 Vlissides 合著的经典书籍《设计模式》为开端，面向对象设计模式的研究和应用成为面向对象程序设计的重要内容。所有结构良好的面向对象软件系统都包含了大量的设计模式，能否熟练应用设计模式已经成为衡量程序员水平的至关重要的标准。

本丛书收录了与 Java 程序设计和 Java 设计模式相关的经典书籍，反映了应用 Java 开发软件系统的最佳经验总结和最新动态。

《Java 设计模式》采用方便而简洁的编写风格，以可视化的 Java 程序为例，详细介绍了 Gamma、Helm、Johnson 和 Vlissides 合著的经典书籍《设计模式》中列出的所有 23 种模式。通过本书，Java 程序员可以迅速了解和掌握设计模式的内容，并在实践中应用设计模式来创建复杂而健壮的 Java 程序。

《Java 模式应用》介绍了基于模式的开发技巧，演示了使用 Java 开发各种商务系统中的模式应用。书中首先概述设计模式，然后就四种主要模式——创建模式、行为模式、结构模式和系统模式展开了详细的论述。该书还针对系统构建过程中常用的 J2EE、JSP、EJB 和 API 等技术作了介绍，适合具有一定编程基础的 Java 程序员阅读参考。

《J2EE 核心模式》是 Sun Java Center 的资深设计师的 J2EE 亲身实践经验的总结。该书主要描述 J2EE 关键技术（Java Server Pages, Java Servlet, Enterprise Java Beans, Java Message Services 等）的模式、最佳实践、设计策略和经过验证的解决方案。该书介绍了 J2EE 包括的 15 个模式的分类和大量的策略，不仅具有理论深度，而且非常实用。该书内容适合 J2EE 的爱好者、程序员、设计师、开发者和技术管理者。一句话，该书

适合于设计、构建和开发 J2EE 平台应用的所有人。

XML 是新一代文档的标准, Web 页、数据、源码等等, 均可以用 XML 文档表示。越来越多的程序员正在使用 Java 来处理 XML 文档。《用 Java 处理 XML》详细论述了如何使用 Java 来读写 XML 文档。该书是目前最新和最全的 Java 处理 XML 技术的介绍, 包含内容超过 1000 页的关于 SAX, DOM, JDOM, JAXP, TrAX, XPath, XSLT, SOAP 等的讲解。该书适合于使用 Java 读写 XML 文档的 Java 程序员。其内容从基本概念到高级应用无所不包, 特别适合作为手册随时参考。

在《实时 Java》中, 作为 RTSJ 专家组的成员之一, Dibble 从 Java 平台特有的实时问题概述开始, 依次讲解了 RTSJ 各项主要特性的使用方法。从广泛的实时原理到详细的编程隐患, 该书覆盖了构建有效实时程序所需的一切知识。其主要内容包括: 与非实时代码的互操作性、实时开发中的取舍以及 JVM 软件的实时问题; 垃圾收集、无堆栈访问、物理内存和“不朽”内存以及无堆栈内存的常数时间分配; 优先级调度、期限调度以及速率单调分析; 闭包、异步传输控制、异步事件以及计时器。这是一本非常实用的指南, 适用于有经验的 Java 平台开发人员。

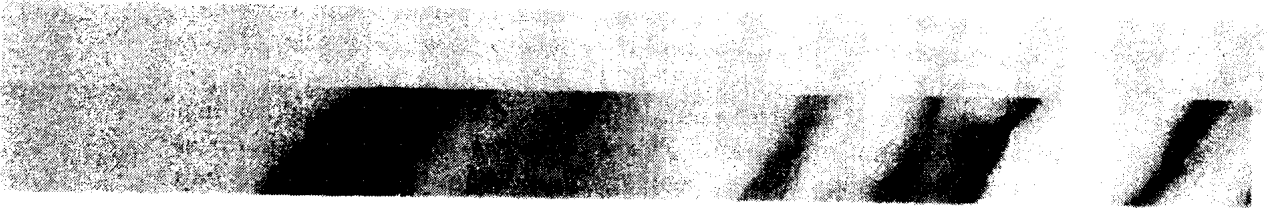
《Java 数据结构与算法分析》介绍了常见的数据结构, 如链表、堆栈、队列、树和哈希表等, 并对查找和排序进行了算法分析, 给出了相应的 Java 实现。该书逻辑结构严谨, 主次分明, 可用作程序员参考书。

总之, 这套书详细介绍了 Java 应用的许多重要方面: 从具有普遍性的 Java 数据结构和算法、Java 设计模式、Java 模式应用、J2EE 核心模式, 到日益显著的 Java 特殊应用领域 (Java 处理 XML 文档和 Java 实时系统开发)。其内容具有一定的理论深度, 更有重要的实际参考价值。

有鉴于此, 特向 Java 系统开发和应用领域中不同程度的读者推荐这套书, 相信每位有心的读者都能得到物超所值的收获。

清华大学经济管理学院管理科学与工程系 朱涛 博士  
讲授课程: Java 程序设计, 面向对象的分析设计方法

# Foreword



In the world of software, a pattern is a tangible manifestation of an organization's tribal memory. A pattern provides a common solution to a common problem and so, within the culture of one specific organization or within one domain, naming and then specifying a pattern represents the codification of a common solution, drawn from proven, prior experience. Having a good language of patterns at your disposal is like having an extended team of experts sitting at your side during development: by applying one of their patterns, you in effect take the benefit of their hard-won knowledge. As such, the best patterns are not so much invented as they are discovered and then harvested from existing, successful systems. Thus, at its most mature state, a pattern is full of things that work, absent of things that don't work, and revealing of the wisdom and rationale of its designers.

Deep, really useful, patterns are typically ancient: you see one and will often remark, "Hey, I've done that before." However, the very naming of the pattern gives you a vocabulary that you didn't have previously and so helps you apply that pattern in ways you otherwise might have not have realized. Ultimately, the effect of such a pattern will be to make your system simpler.

Patterns not only help you build simpler systems that work, but they also help you build beautiful programs. In a culture of time

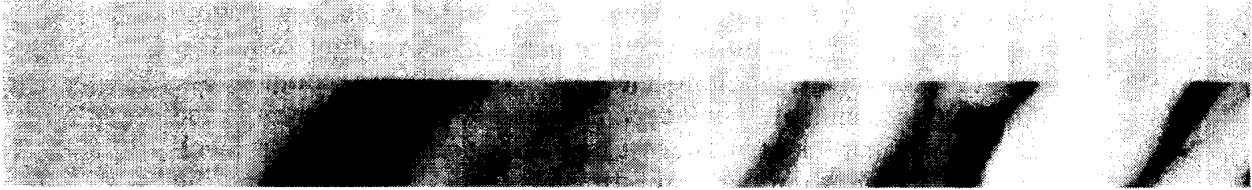
starvation, writing beautiful software is often impossible. That's sad, for as professionals, we strive to build things of quality. By applying a good set of patterns, it is possible to bring a degree of elegance in to your systems that might otherwise have been lacking.

The authors of *Core J2EE Patterns* have harvested a really useful set of patterns. Don't get me wrong: J2EE is certainly an important platform, enabling teams to build some very powerful systems. However, reality is, there is still a wide semantic gap between the abstractions and services that J2EE provides and the final application that a team must build. Patterns such as specified in this book represent solutions that appear again and again in filling that gap. By applying these patterns, you thus carry out the primary means of reducing software risk: you write less software. Rather than discovering these solutions on your own, apply these patterns, which have already proven their utility in existing systems.

More than just naming a set of patterns, the authors make them approachable by specifying their semantics using the UML. Additionally, they show you how to apply these patterns and how to refactor your system to take advantage of them. Again, it's just like having a team of experts sitting at your side.

Grady Booch  
Chief Scientist  
Rational Software Corporation

# Preface



This book is about patterns for the Java 2 platform, Enterprise Edition (J2EE). These J2EE patterns provide solutions for problems typically encountered by designers of software applications for the J2EE platform. All the patterns documented in the catalog have been discovered in the field, where they have been used to create successful J2EE applications for our customers.

This book describes proven solutions for the J2EE platform with a particular emphasis on such key J2EE technologies as: Java Server Pages (JSP), Servlets, Enterprise JavaBeans (EJB) components, Java Message Service (JMS), JDBC, and Java Naming and Directory Interface (JNDI). We offer solutions for recurring problems for the J2EE platform through the J2EE Pattern Catalog and J2EE refactorings. You can apply these ideas when developing new systems or when improving the design of existing systems. The patterns in this book will help you quickly gain the proficiency and skills to build robust, efficient enterprise applications.

Today, as in the past, many of us naively assume that *learning a technology* is synonymous with *learning to design* with the technology. Certainly learning the technology is an important part to being successful in designing with the technology. Many existing Java books are excellent at explaining technology details, such as API specifics and so forth, but at the same time they give no insight on



applying the technology. Learning to design comes from experience and from sharing knowledge on best practices and bad practices.

The experiences we have conveyed in this book are derived from the work we have done in the field. We are part of Sun Microsystems, Inc.'s Sun Java Center (SJC) consulting organization. In our work, we often encounter situations where, because technology is moving so quickly, designers and developers are still struggling to understand the technology, let alone how to design with the technology.

It is not good enough to tell designers and developers to write good code, nor is it sufficient to suggest using Servlets and JSP for developing the presentation tier and EJB components<sup>1</sup> for developing the business tier.

So, given this scenario, where does an aspiring J2EE architect learn not only what to do, but what not to do? What are the best practices? What are the bad practices? How do you go from problem to design to implementation?

## Sun Java Center and the J2EE Pattern Catalog

Since its inception, SJC architects have been working with clients all over the world to successfully design, architect, build, and deploy various types of systems based on Java and J2EE. The SJC is a rapidly growing consulting organization constantly adding new hires to its ranks of experienced architects.

Recognizing the need to capture and share proven designs and architectures, we started to document our work on the J2EE platform in the form of patterns in 1999. Although we looked in the existing literature, we could not find a catalog of patterns that dealt specifically with the J2EE platform. We found many books dealing with one or more of the J2EE technologies, and these books do an excellent job of explaining the technology and unraveling the nuances of the specifications. Some books offered extra help by providing some design considerations.

---

1. If you are new to the J2EE platform, we discuss the platform and these technologies in Chapter 2, "J2EE Platform Overview".

Since we first publicly presented our ideas on J2EE patterns at the JavaOne Conference in June 2000, we have received an overwhelming response from architects and developers. While some individuals expressed great interest in learning more about the patterns, others confirmed that they had applied the patterns, but had never named or documented them. This interest in patterns for the J2EE platform further motivated us to continue our work.

Thus, we put together the J2EE Pattern Catalog, which was initially made available to the entire J2EE community in beta form via the Java Developer Connection in March, 2001. Based largely on community feedback, the beta documentation evolved into the release you see in this book.

We hope these patterns, best practices, strategies, bad practices, and refactorings for the J2EE platform, provide the same benefits to you as they do for us.

## What This Book is About?

This book is about:

- *Using patterns for the J2EE Platform.*  
Based on our collective J2EE platform experience, we have assembled the pattern catalog in this book. The J2EE Pattern Catalog describes various best practices related to architecting and designing applications for the J2EE platform. This book focuses on the following four J2EE technologies: Servlets, JSP, EJB components, and JMS.
- *Using best practices to design applications that use JSP, Servlet, EJB components, and JMS technologies.*  
It is not sufficient to merely learn the technology and the APIs. It is equally important to learn to design with the technology. We have documented what we have experienced to be the best practices for these technologies.
- *Preventing re-inventing-the-wheel when it comes to design and architecture for the J2EE platform.*  
Patterns promote design reuse. Reusing known solutions

reduces the cycle time for designing and developing applications, including J2EE applications.

- *Identifying bad practices in existing designs and refactoring these designs to move to a better solution using the J2EE patterns.*

Knowing what works well is good. Knowing what does not work is equally important. We have documented some of the bad practices we have experienced when designing applications for the J2EE platform.

## What This Book Is Not?

This book is not about:

- *How to program with Java or J2EE technologies*  
This book is not about programming. While this book is heavily based on the J2EE technologies, we do not describe the specific APIs. If you wish to learn about programming using Java or using any of the J2EE technologies, there are a number of excellent books and online resources from which to learn. The online tutorials on the official Java home page at <http://java.sun.com> are highly recommended if you wish to learn about individual technologies. The official specifications for J2EE technologies are also available from the Java home page.
- *What process and methodology to use*  
We do not suggest any type of process or methodology to use since the material presented in this book is not related to either. Hence, this book does not teach you about a process or methodology to follow in your projects. If you would like to learn more about processes and methodologies, there are a good number of books that deal with various object-oriented methodologies and new books on lightweight processes, such as Extreme Programming.
- *How to use Unified Modeling Language (UML)*  
This book is not going to teach you about UML. We use

UML extensively (specifically class and sequence diagrams) to document the patterns and describe the static and dynamic interactions. If you want to learn more about UML, please refer to the UML User Guide [Booch] and the UML Reference Manual [Rumbaugh] by Grady Booch, Ivar Jacobson and James Rumbaugh.

## Who Should Read this Book?

This book is for all J2EE enthusiasts, programmers, architects, developers, and technical managers. In short, anyone who is remotely interested in designing, architecting and developing applications for the J2EE platform.

We have attempted to distinguish this book as a training guide for J2EE architects and designers. We all recognize the importance of good designs and well-architected projects, and that we need good architects to get there.

The use of well-documented patterns, best practices, and bad practices to share and transfer knowledge and experience can prove invaluable for teams with varied experience levels, and we hope that this book answers some of these needs.

## How This Book is Organized

This book is organized into three parts.

Part 1—"Patterns and J2EE", consists of Chapter 1 and Chapter 2.

Chapter 1: "Introduction" on page 4 is a brief discussion on various topics, including patterns, J2EE platform, defining a pattern, and pattern categorization. It ends by introducing the J2EE Pattern Catalog.

Chapter 2 : "J2EE Platform Overview" on page 16 provides a high level overview of the J2EE platform for those readers unfamiliar with J2EE, or who wish to refresh their knowledge of the J2EE platform.

Part 2—"Design Considerations, Bad Practices, and Refactorings" deals with design considerations for JSP, Servlets, and enterprise beans. This part also includes bad practices and refactorings for the J2EE platform. This part is comprised of Chapter 3, 4, and 5.



## Preface

Chapter 3 “Presentation Tier Design Considerations and Bad Practices” on page 34 and Chapter 4 “Business Tier Design Considerations and Bad Practices” on page 54 discuss the design considerations and bad practices for the presentation tier and business/integration tiers respectively. The design considerations are issues that a J2EE developer/designer/architect needs to consider while working with the J2EE platform. The topics presented in these chapters point the reader to other sources (such as official specifications and well written books on these topics) for more detailed information on these issues.

Chapter 5: “J2EE Refactorings” on page 72 includes some of the refactorings we have experienced in our work in the field that has enabled us to move our design from a less optimal solution to a better solution. The refactorings provide another way to think about the material in the rest of the book, providing what we believe to be valuable companion material to the pattern catalog. This chapter shows how we have been influenced by Martin Fowler and his book “Refactoring” [Fowler]. For those readers who are familiar with the Refactoring book, the format of this chapter will be very familiar. However, the content of this chapter is entirely in the context of J2EE technologies, whereas Martin Fowler addresses refactoring at a different level.

Part 3—“J2EE Pattern Catalog” presents the J2EE pattern catalog. The catalog contains the fifteen patterns that form the core of this book. This part is comprised of Chapter 6, 7, 8, and 9.

Chapter 6: “J2EE Patterns Overview” on page 124 provides an overview of the J2EE pattern catalog. This chapter begins with a high level discussion of the pattern ideas and explains the way the patterns are categorized into tiers. It also explains the J2EE pattern template, which is used to present all patterns in this book. The chapter discusses all the J2EE patterns and uses a diagram to show their inter-relationships. It also provides what we have termed a roadmap to the pattern catalog. This roadmap presents common J2EE design and architecture-related questions with references to patterns or refactorings that provide solutions to these questions. Understanding the pattern relationships and the roadmap is key to using these patterns.

Chapter 7: “Presentation Tier Patterns” on page 150 presents six patterns that pertain to using Servlets, JSP, JavaBeans, and custom tags to design web-based applications for the J2EE platform. The

patterns describe numerous implementation strategies, and address common problems such as request handling, application partitioning, and generating composite displays.

Chapter 8: “Business Tier Patterns” on page 246 presents seven patterns that pertain to using EJB technology to design business components for the J2EE platform. The patterns in this chapter provide the best practices for using the EJB and JMS technologies. Where relevant, these patterns include discussion on other technologies, such as JNDI and JDBC.

Chapter 9: “Integration Tier Patterns” on page 388 presents two patterns that pertain to integrating J2EE applications with the resource tier and external systems. The patterns deal with using JDBC and JMS to enable integration between business tier and resource tier components.

Epilogue: “J2EE Patterns Applied” on page 422 discusses realizing sample use cases with the patterns. This chapter discusses and demonstrates how patterns are combined and work together. This chapter reinforces the idea that patterns exist in a community, and that each pattern supports, and is supported by, other patterns.

## **Companion Website and Contact Information**

The official companion website where we will provide updates and other material is <http://www.phptr.com/corej2eepatterns>.

The J2EE Patterns interest group, [j2eepatterns-interest@java.sun.com](mailto:j2eepatterns-interest@java.sun.com) is available for public subscription and participation. To subscribe to the interest group and review the discussion archives, please visit:

<http://archives.java.sun.com/archives/j2eepatterns-interest.html>

## Acknowledgments

We wish to thank Stu Stern, Director of Global Sun Java Center and Mark Bauhaus, VP of .COM Consulting without whose support, vision, and belief in our work this effort would never have been realized.

We wish to thank Ann Betser, without whose support, encouragement and skilled advice, we would have been lost.

We wish to express our sincere thanks to the PSA/iWorkflow reference implementation team of SJC architects: Fred Bloom, Narayan Chintalapati, Anders Eliasson, Kartik Ganeshan, Murali Kalyanakrishnan, Kamran Khan, Rita El Khoury, Rajmohan Krishnamurthy, Ragu Sivaraman, Robert Skoczylas, Minnie Tanglao, and Basant Verma.

We wish to thank the Sun Java Center J2EE Patterns Working Group members: Mohammed Akif, Thorbiörn Fritzson, Beniot Garbinato, Paul Jatkowski, Karim Mazouni, Nick Wilde, and Andrew X. Yang.

We wish to thank Brendan McCarthy, SJC Chief Methodologist for keeping us in balance and for all the advice.

We wish to thank Jennifer Helms and John Kapson for introducing the patterns to customers.

We wish to express our gratitude to the following SJC architects from around the world for their support, feedback, and advice: Mark Cade, Mark Cao, Torbjörn Dahlén, Peter Gratzner, Bernard Van Haecke, Patricia de las Heras, Scott Herndon, Grant Holland, Girish Ippadi, Murali Kaundinya, Denys Kim, Stephen Kirkham, Todd Lasseigne, Sunil Mathew, Fred Muhlenberg, Vivek Pande, John Prentice, Alexis Roos, Gero Vermaas, Miguel Vidal.

We wish to thank our management Hank Harris, Dan Hushon, Jeff Johnson, Nimish Radia, Chris Steel, and Alex Wong for their support and encouragement.

We wish to thank the following Sun colleagues for their collaboration:

Bruce Delagi from Software Systems group; Mark Hapner, Vlada Matena from Java Software Engineering; Paul Butterworth and Jim Dibble from Forte Products Group; Deepak Balakrishna from iPlanet Products Group; Larry Freeman, Cori Kaylor, Rick Saletta, and Inderjeet Singh from the J2EE Blueprints Team; Heidi Dailey; Dana

Nourie, Laureen Hudson, Edward Ort, Margaret Ong, and Jenny Pratt from Java Developer Connection.

We wish to thank the following for their feedback, advice, and support:

Martin Fowler and Josh Mackenzie from ThoughtWorks, Inc.; Richard Monson-Haefel; Phil Nosonowitz and Carl Reed from Goldman Sachs; Jack Greenfield, Wojtek Kozaczynski, and Jon Lawrence from Rational Software; Alexander Aptus from TogetherSoft; Kent Mitchell from Zaplets.com; Bill Dudney; David Geary; Hans Bergsten; Members of the J2EE Patterns Interest group ([j2eepatterns-interest@java.sun.com](mailto:j2eepatterns-interest@java.sun.com)).

We wish to express our special thanks and gratitude to our lead technical editor Beth Stearns, transforming our manuscripts and making them readable, at the same time keeping us on track, and working with us all the way with a heavily demanding schedule.

We wish to thank the technical editors Daniel S. Barclay, Steven J. Halter, Spencer Roberts, and Chris Taylor for their expertise, meticulous review and feedback.

We wish to thank Greg Doench, Lisa Iarkowski, Mary Sudul, and Debby Van Dijk from Prentice Hall; Michael Alread and Rachel Borden from Sun Microsystems Press, for doing everything it took to produce this book.

We thank Bill Jirsa, John Hathaway, and Darlene Khosrowpour from Sun Educational Services for their effort creating the SunEd J2EE Patterns course (SL-500), John Sharp and Andy Longshaw from Content Master Ltd., as well as all the course reviewers for SL-500.

We wish to thank the patterns and the Java communities on whose work we have built.

The authors wish to thank their families for their support.

Deepak Alur wishes to thank:

Kavya, Shivaba and Samiksha—for your support, understanding, and inspiration; My Parents and Ajay.

John Crupi wishes to thank:

Ellen and Rachel—for your support, understanding and love.  
Casey and Smokey—two great dogs will be forever missed.



**Preface**

Dan Malks wishes to thank:

Beth, Sarah, and Jonathan—for your support and for bringing special meaning to everything in my life.