



软件工程技术

UML和模式应用

面向对象分析与设计导论

Applying UML and Patterns
An Introduction to
Object-Oriented Analysis and Design



(美) Craig Larman

姚淑珍 李虎 译

软件工程技术丛书

UML 和模式应用

面向对象分析与设计导论

(美) Craig Larman 著

姚淑珍 李 虎 等译



机械工业出版社
China Machine Press

本书论述运用 UML (统一建模语言) 和模式进行对象建模的方法和技巧, 重点讨论了如何使用面向对象的分析和设计技术来建造一个健壮的和易于维护的系统。

全书叙述清晰、图文并茂、实例丰富, 是一部来自于大量经验的总结性论著, 适合在学习和工作中需要运用面向对象技术的高校师生或工程技术人员使用, 特别适用于对面向对象技术有一定了解但希望进一步提高开发水平的应用开发人员。

Craig Larman: Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design.

Authorized translation from the English language edition published by Prentice Hall PTR. Copyright © 1998 by Craig Larman.

All rights reserved.

Chinese simplified language edition published by China Machine Press.

Copyright © 2002 by China Machine Press.

本书中文简体字版由美国 Prentice Hall PTR 公司授权机械工业出版社独家出版。未经出版者书面许可, 不得以任何方式复制或抄袭本书内容。

版权所有, 侵权必究。

本书版权登记号: 图字: 01-2000-4302

图书在版编目 (CIP) 数据

UML 和模式应用: 面向对象分析与设计导论 / (美) 拉尔曼 (Larman, C.) 著; 姚淑珍等译. - 北京: 机械工业出版社, 2002.1

(软件工程技术丛书)

书名原文: Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design

ISBN 7-111-09358-5

I . UML... II . ①拉...②姚... III . 面向对象语言, UML—程序设计 IV . TP312

中国版本图书馆 CIP 数据核字 (2001) 第 065360 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: 姚 蕾

北京市密云县印刷厂印刷 · 新华书店北京发行所发行

2002 年 1 月第 1 版 · 2002 年 5 月第 2 次印刷

787mm×1092mm 1/16 · 26 印张

印数: 5 001-8 000 册

定价: 48.00 元

凡购本书, 如有倒页、脱页、缺页, 由本社发行部调换

译者序

面向对象方法以其超越传统方法的技术先进性越来越得到更多的重视，但技术的先进性不能完全代表一种新生事物的最后成功，还要看它是否能得到有效的推广。统一建模语言(UML)的产生为这种推广规定了一致的表示，诸多开发过程模型为技术的应用提供了宏观的指导。但开发者往往会说：“面对一个应用，我了解UML，也知道应基于用况，以体系结构为核心，采用迭代、增量模式，来实施开发过程。但我仍困惑，从何入手，具体该怎么做？”如果您也有这样疑问的话，本书将为您提供开发过程和设计方法的指导。

本书给出了一个开发过程示例，它描述了一个可能的活动序列和开发周期。通过一个学习案例——销售点终端系统的开发过程引导，尽可能实际地展示面向对象的分析和设计过程，以及所面临的问题及其解决办法。本书还通过讲授GRASP等典型模式帮助您有效学习和使用面向对象分析与设计的实用技巧。这些技巧是使用对象技术和面向对象编程语言（例如C++、Java或Smalltalk）开发出设计良好、结构健壮和易于维护的软件系统所必不可少的。

本书是一部来自于大量经验的总结性论著，服务对象是学习或工作中运用面向对象技术的教师、学生或工程技术人员，特别适用于对面向对象技术有一定了解但希望进一步提高开发水平的应用开发人员。

本书由姚淑珍、李虎负责翻译，胡斌、姚志强、黄涛、黄昕、许正华、张翼翻译了部分内容。大家集思广益、畅所欲言、反复推敲，形成了对原文的基本一致和较透彻的理解。

在翻译中我们尽量融入了多年来从事软件工程的经验和面向对象开发的实践体会，但也难免存在由于知识局限和时间仓促的原因造成的翻译错误和遗漏，敬请广大读者指正。我们会接受您的批评，并向原书作者表示歉意。

译者

2002年1月于北京

译者介绍

姚淑珍（女），北京航空航天大学计算机科学与工程系副教授。1989年获硕士学位。研究方向为软件工程、面向对象技术、分布式计算。已在国内外重要期刊和学术会议发表论文30余篇，并获多项国家及部委科技成果奖和个人奖。

李虎（男），北京航空航天大学计算机系博士生，研究方向为软件工程、Agent技术、面向对象技术，发表论文近10篇。

前 言

感谢你阅读本书！现在你的手里已经有了这本面向对象的分析与设计世界的实用指南和引路图了。下面要说明本书将如何使你受益。

首先，帮助你设计健壮的和易于维护的对象系统。在软件开发过程中对象技术的使用日益增多——甚至随着 Java 语言的广泛采用，这种对象技术的使用还会更多——掌握面向对象的分析技术对建立一个健壮的和易于维护的面向对象系统来说是至关重要的。掌握这项技术同样也为你开辟了成为构架设计师、分析人员和设计人员的新机会。

第二，提供从需求确定、系统分析、设计到编码这个过程的引路图。如果你是一位面向对象分析和设计方面的新手，那你一定会面对这样一个难题：如何进行面向对象的分析和设计这一复杂课题？本书为你提供了一个表达准确的引路图，以使你能够一步一步地从需求分析开始一直走到程序编码。

第三，教你运用 UML 描述分析和设计模型。统一建模语言（Unified Modeling Language, UML）是作为一种标准的建模表示法应运而生的，因此掌握它对你大有用处。本书将为你讲授使用 UML 表示法进行面向对象分析与设计的技巧。

第四，运用“四人”模式（gang-of-four）和 GRASP 模式改进设计。设计模式是面向对象分析方面的专家们所总结的“最好的做法”。本书将教你使用设计模式，包括目前流行的“四人模式”，以及最为重要的 GRASP 模式。GRASP 模式能够反映出面向对象设计中有关职责分配方面的基本原则和方法。学习和运用模式技术会有助于你更快地掌握分析和设计技术。

第五，通过一套精细的表述方式进行卓有成效地学习。全书的组织结构和重点内容是基于作者多年培养和训练人们使用面向对象分析和设计技巧所积累的经验总结。本书通过提供一套精练的、被证明是正确的和高效率的方法来使这些经验结晶能够在书中体现出来，以使你对本书的投资和阅读能够得到最佳的回报。

第六，通过符合实际的例子辅助学习。书中详细描述了一个学习案例——以便尽可能符合实际地展示完整的面向对象的分析和设计过程。本书还深入细致地探讨了问题的各个方面细节。这个学习案例是一个符合实际的练习题。

第七，学会把设计转换成代码。本书展示了如何将面向对象的设计制品转换成 Java 语言程序代码的方法。

第八，设计出一种分层的系统体系结构。本书阐述了如何设计分层的系统体系结构以及如何将图形用户界面层与应用领域和系统服务层联系起来。这个问题在实际应用中意义重大但却往往会被人们所忽视。

最后，设计出系统框架。书中还介绍了如何设计面向对象的框架，并特别介绍了如何将这种设计方法应用于框架的创建以便用于数据库中的持久化存储。

目标

本书的主要目标是：

通过应用一套可以解释的设计原则和启发式的教学法帮助学生和开发人员创建更好的面向对象的设计。

通过学习和运用本书所提供的信息和技术，你将能够更加熟练地从概念和过程两个方面来理解问题，并能够运用对象技术设计更加完善的问题解决方案。

适用读者

本书适用于以下读者：

- 对面向对象程序设计语言有一定使用经验，但还不熟悉或不很熟悉面向对象分析与设计技巧的开发人员。
- 在计算机科学或软件工程课程中学习对象技术的学生。
- 对面向对象分析与设计有一定熟悉程度，并想进一步学习统一建模语言表示法和运用设计模式的人，或者想磨练和提高自己的分析设计技巧的人。

预备知识

为了更好地从本书受益，读者必须具备以下一些预备知识：

- 面向对象的程序设计语言（如 C++、Java 或 Smalltalk）方面的知识和使用经验。
- 了解对象技术的基本概念，如类、实例、接口、多态、封装和继承等。

本书中没有给出上述的对象技术中最基本概念的定义。

本书的组织

本书的基本组织思路是：以类似于软件项目开发所经历的两个迭代开发周期的顺序来引入面向对象分析与设计的各个专题。第一个迭代开发周期中引入了一定的分析与设计。第二个迭代开发周期又引入了新的分析和设计专题，与此同时，还将对前一个周期中的专题进行更深入的讨论。见图 1 所示。

本书目的

尽管对象技术很有发展前景，但是如果不能正确地使用某些技巧的话，人们将很难意识到对象技术所具有潜力。我的目标是通过运用面向对象分析与设计技巧来促进对象技术的成功应用，以及促进人们在这方面技术能力的提高。我这样做的原因在于我已经觉察到我所提到的

这些技巧是成功地开发和维护重要系统项目的关键因素。

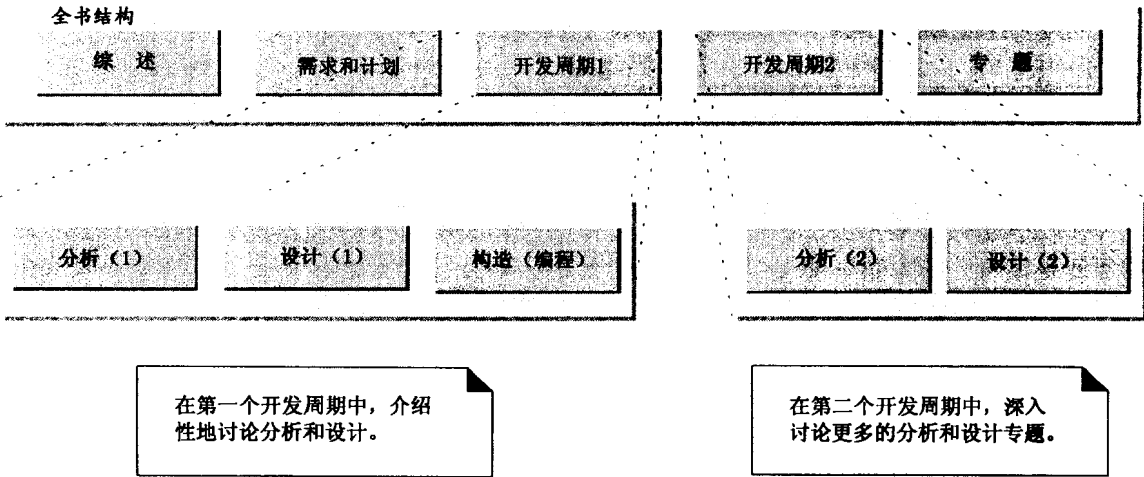


图 1 按照项目开发过程来组织本书的内容

致谢

首先要感谢所有使用、学习 UML 和面向对象分析与设计技术的人, 我写这本书正是要帮助他们, 同时他们也是我最好的老师。

这里要特别感谢本书 (或其中部分章节) 的审阅者, 包括 Kent Beck、Jens Coldewey、Clay Davis、Tom Heruska、Luke Hohmann、David Norris、David Nunn、Brett Schuchert 以及整个水星组 (Mercury team)。感谢 Grady Booch 对本书相关教学材料的审阅。还要感谢 Jim Rumbaugh, 他就书中 UML 和软件过程之间的关系等内容提出了很多反馈意见。

因为还有许多人对书中有关模型和过程等内容提出了宝贵意见, 所以在此我还要感谢 Todd Girvin、John Hebley、Tom Heruska、David Norris、David Nunn、Charles Rego 和 Raj Wall 等。

非常感谢 Grady Booch、Ivar Jacobson 和 Jim Rumbaugh。正是他们三位合作开发了统一建模语言并创建了一个标准的、开放的模型表示法, 这在当前无章可循的大环境下是深受业界欢迎的。此外, 我还从他们的主张中学到了很多东西。

感谢我的同事 Jef Newsom 为本书增加了 Java 解决方案的案例学习内容。

感谢我的出版商 Prentice-Hall 公司的 Paul Becker, 正是他坚信本书是值得出版的好书。

最后, 谢谢 Graham Glass 提供的一切便利。

关于作者

作者拥有计算机科学学士和硕士学位, 他从 1978 年开始应用从 4GL 编程、逻辑程序设计

到面向对象程序设计等各种软件技术，从事从大型计算机到微机等各种平台上的大型软件系统和小型软件系统的开发。

在 20 世纪 80 年代初期，作者钟情于人工智能和知识系统中的程序设计技术，并从那时起开始接触到面向对象的程序设计（使用 Lisp 语言）。他从 1984 年开始使用和讲授 Lisp 语言的面向对象程序设计，到 1986 年改用 Smalltalk，到 1991 年变为 C++，最近又开始使用 Java 语言，并结合程序设计讲授各种面向对象的分析和设计方法。他在对象技术领域指导和帮助过的学生超过 2000 人。

作者现任 ObjectSpace 公司的首席讲师，ObjectSpace 是一家专门研究分布式计算、agent 技术和对象技术的公司。

读者可以通过电子邮箱：clarman@acm.org 与作者本人联系。

排版约定

新出现的术语用黑体印刷。

类名或者方法名如果保持英文原文用斜体印刷，如果译为中文用楷体印刷。

相关作者的参考文献用方括号括住的文献缩写来引用，如 [Bob67]。

“--”符号用作类名和类中方法名之间的连接符，如 *ClassName--methodName()*。

目 录

译者序 前 言

第一部分 绪论

第 1 章 面向对象的分析与设计	3
1.1 运用 UML、模式和面向对象的分析与设计技术	3
1.2 分配职责	4
1.3 什么是分析和设计	5
1.4 什么是面向对象的分析和设计	5
1.5 类比——组织 MicroChaos 公司的业务	6
1.5.1 MicroChaos 公司正迅速发展壮大	6
1.5.2 什么是业务过程	6
1.5.3 组织中的角色是什么	7
1.5.4 谁该干什么？他们之间如何协作	7
1.6 面向对象的分析与设计的例子	8
1.6.1 定义用况	8
1.6.2 定义概念模型	9
1.6.3 定义协作图	9
1.6.4 定义设计类图	10
1.6.5 掷骰子游戏例子的总结	10
1.7 面向对象的与面向功能的分析与设计	11
1.8 警告：“分析”和“设计”可能引起术语上的“冲突”	12
1.9 统一建模语言	12
第 2 章 开发过程导论	14
2.1 导言	14
2.1.1 推荐的过程和模型——RPM	14
2.1.2 讨论范围	15
2.2 UML 和开发过程	15

2.3 高层步骤	16
2.4 迭代开发	16
2.4.1 确定开发周期的时间盒	16
2.4.2 用况和迭代开发周期	17
2.4.3 划分用况的层次	18
2.5 计划和细化阶段	18
2.6 构造阶段——开发周期	19
2.7 选择制品创建的时机	20
2.7.1 何时创建概念模型	20
2.7.2 何时创建扩展用况	21
第 3 章 定义模型和制品	23
3.1 导言	23
3.2 建模系统	23
3.3 样例模型	23
3.4 制品之间的关系	24

第二部分 计划和细化阶段

第 4 章 学习案例：销售点终端	29
4.1 销售点终端系统	29
4.2 系统体系结构的层次和学习案例的重点	29
4.3 我们的策略：反复学习和反复开发	30
第 5 章 理解需求	32
5.1 导言	32
5.2 需求	33
5.3 总体问题陈述	34
5.4 顾客	34
5.5 目标	34
5.6 系统功能	34
5.6.1 功能的分类	34
5.6.2 基本功能	35
5.6.3 处理支付的功能	35
5.7 系统属性	35
5.8 需求阶段的其他制品	36

第6章 用况：对过程的描述	38
6.1 引言	38
6.2 活动及其相互间的依赖关系	39
6.3 用况	39
6.3.1 高层用况举例：购买商品	39
6.3.2 扩展用况举例：用现金购买 商品	40
6.3.3 扩展格式的说明	41
6.4 参与者	41
6.5 使用用况时的常见错误	42
6.6 用况的识别	42
6.7 用况和领域过程	43
6.8 用况、系统功能和可跟踪性	43
6.9 用况图	43
6.10 用况的格式	44
6.10.1 高层格式	44
6.10.2 扩展格式	44
6.11 系统及其边界	44
6.12 主要、次要和可任选的用况	46
6.13 基本用况和真实用况	46
6.13.1 基本用况	46
6.13.2 真实用况	47
6.13.3 购买商品的基本用况	47
6.13.4 购买商品的真实用况	47
6.14 表示法要点	48
6.14.1 用况的命名	48
6.14.2 扩展用况的开始部分	48
6.14.3 判定点和分支的表示法	48
6.15 一个开发过程中的用况	49
6.15.1 计划和细化阶段的步骤	49
6.15.2 迭代开发周期阶段中的步骤	50
6.16 销售点终端系统的处理步骤	50
6.16.1 识别参与者和用况	50
6.16.2 用高层格式书写用况	51
6.16.3 绘制系统的用况图	51
6.16.4 为用况增加关联	52
6.16.5 书写一部分扩展的基本用况	52
6.16.6 在必要情况下书写一些真实 用况	54
6.16.7 划分用况的层次	54

6.17 样例模型	54
第7章 用况的分类和时间调度	55
7.1 引言	55
7.2 将用况调度分配到开发周期中实现	56
7.2.1 用况和开发周期	56
7.2.2 用况的分类	56
7.3 销售点终端应用系统中的用况分类	57
7.4 “系统启动”用况	58
7.5 销售点终端应用系统中用况的时间 调度	58
7.5.1 创建复杂用况的多个版本	58
7.5.2 用况分配	58
7.6 “购买商品”用况的版本	58
7.6.1 购买商品—版本1	59
7.6.2 购买商品—版本2	60
7.7 小结	60
第8章 开始进入一个开发周期	61

第三部分 分析阶段 (1)

第9章 建立一个概念模型	65
9.1 引言	65
9.2 活动及其相互之间的依赖关系	66
9.3 概念模型	66
9.3.1 理解领域词汇	67
9.3.2 概念模型不是软件设计模型	67
9.3.3 概念	68
9.3.4 概念模型和问题分解	68
9.3.5 销售点终端系统问题域中的 概念	69
9.4 识别概念的策略	69
9.4.1 使用概念目录列表来找出概念	70
9.4.2 根据名词性短语找出概念	71
9.5 销售点终端问题域中的候选概念	71
9.5.1 报告类对象——模型中包括 收据吗	72
9.5.2 销售点终端系统的概念模型（只包 括概念）	72
9.6 建立概念模型的指导原则	72
9.6.1 怎样建立一个概念模型	72
9.6.2 事物的命名和建模：制图者的	

方法	73	11.4.3 纯数据值	90
9.6.3 在识别概念时常犯的错误	73	11.4.4 设计问题蔓延: 避免使用 外部键属性	91
9.7 类似概念的解析——POST 和 Register	74	11.5 非简单属性类型	91
9.8 非现实世界中的概念建模	75	11.6 对属性的数量和单位建模	93
9.9 规格说明或描述型概念	75	11.7 销售点系统中的属性	93
9.9.1 对规格说明的需求	75	11.8 销售点模型中的属性	94
9.9.2 什么时候需要规格说明型 概念	76	11.9 从 <i>SalesLineItem</i> 到 <i>Item</i> 的多重性	95
9.9.3 另一个规格说明的例子	76	11.10 销售点系统的概念模型	95
9.10 UML 中有关术语的定义	77	11.11 小结	96
9.11 样例模型	78	第 12 章 在术语表中记录术语	97
第 10 章 概念模型——添加关联	79	12.1 导言	97
10.1 导言	79	12.2 术语表	97
10.2 关联	79	12.3 活动及其相互之间的依赖关系	97
10.3 关联的 UML 表示法	80	12.4 销售点系统的术语表的示例	97
10.4 找出关联——通用关联列表	80	第 13 章 系统行为——系统顺序图	99
10.5 关联应该精细到什么程度	81	13.1 导言	99
10.6 关联原则	81	13.2 活动及其相互之间的依赖关系	99
10.7 角色	82	13.3 系统行为	100
10.8 关联的命名	83	13.4 系统顺序图	100
10.9 两个类型间的多重关联	83	13.5 系统顺序图的例子	101
10.10 关联和它的实现	84	13.6 系统事件和系统操作	102
10.11 销售点问题域中的关联	84	13.7 如何建立一个系统顺序图	103
10.11.1 在商店中不能被遗忘的一些 关系	84	13.8 系统顺序图和其他制品	103
10.11.2 使用关联核对列表的分类	85	13.9 系统事件和系统边界	103
10.12 销售点系统的概念模型	85	13.10 系统事件和操作的命名	105
10.12.1 只保存“需要知道”型 关联吗	85	13.11 显示出用况的文本描述	105
10.12.2 “需要知道”型关联和“理解” 型关联	87	13.12 样例模型	105
第 11 章 概念模型——添加属性	88	第 14 章 系统行为——契约	107
11.1 导言	88	14.1 导言	107
11.2 属性	88	14.2 活动及其相互之间的依赖关系	107
11.3 属性的 UML 表示法	88	14.3 系统行为	108
11.4 有效的属性类型	89	14.4 契约	108
11.4.1 保持属性的简单性	89	14.5 契约举例—— <i>enterItem</i>	109
11.4.2 分析和设计相比: 属性如何 用代码来实现	90	14.6 契约段	109
		14.7 如何建立一个契约	110
		14.8 后置条件	111
		14.8.1 后置条件与概念模型相关	111
		14.8.2 使用后置条件的优点	111
		14.9 后置条件的核心: 舞台和帷幕	111

14.10 讨论—— <i>enterItem</i> 的后置条件	112	17.9.5 返回值的表示法	131
14.10.1 实例的创建和销毁	112	17.9.6 消息的语法	131
14.10.2 属性的修改	112	17.9.7 传送到“self”或“this”的消息 表示法	132
14.10.3 关联的形成和破裂	112	17.9.8 迭代的表示法	132
14.11 后置条件应该详细到什么程度	113	17.9.9 实例创建的表示法	133
14.12 描述设计细节和算法——注释	113	17.9.10 消息序号的表示法	134
14.13 前置条件	113	17.9.11 条件消息的表示法	135
14.14 对书写契约的一些建议	113	17.9.12 互斥条件路径的表示法	136
14.15 用况 <i>enterItem</i> 的契约	114	17.9.13 实例集的表示法	136
14.15.1 <i>enterItem</i> 的契约	114	17.9.14 传递给多对象的消息表示法	137
14.15.2 <i>endSale</i> 的契约	115	17.9.15 发送给类对象的消息表示法	138
14.15.3 <i>makePayment</i> 的契约	115	17.10 样例模型	138
14.16 用况 <i>StartUp</i> 的契约	116	第 18 章 GRASP: 职责分配模式	139
14.17 概念模型的修改	116	18.1 引言	139
14.18 样例模型	117	18.2 活动及其相互之间的依赖关系	140
第四部分 设计阶段 (1)		18.3 设计优良的交互图很有价值	140
第 15 章 从分析到设计	121	18.4 职责和方法	141
15.1 分析阶段的总结	121	18.5 职责和交互图	141
15.2 设计阶段的开始	121	18.6 模式	142
第 16 章 描述真实用况	122	18.6.1 模式通常不包含新的设计 思想	142
16.1 引言	122	18.6.2 模式带有名称	143
16.2 活动及其相互之间的依赖关系	122	18.6.3 模式被命名后有利于增进 交流	143
16.3 真实用况	122	18.7 GRASP: 职责分配中通用原则的 模式	143
16.4 举例——购买商品—版本 1	123	18.8 UML 类图表示法	144
16.5 样例模型	124	18.9 专家	144
第 17 章 协作图	125	18.10 创建者	148
17.1 引言	125	18.11 低耦合度	150
17.2 活动及其相互之间的依赖关系	125	18.12 高聚合度	152
17.3 交互图	126	18.13 控制者	154
17.4 协作图举例: <i>makePayment</i>	127	18.14 职责、角色扮演和 CRC 卡	162
17.5 交互图是一个很有价值的制品	128	第 19 章 运用对象和模式设计一个解决 方案	163
17.6 本章只介绍表示法	128	19.1 引言	163
17.7 阅读后面的章节来学习设计原则	128	19.2 交互图和其他制品	163
17.8 如何建立协作图	128	19.2.1 交互图和系统事件	164
17.9 协作图的基本表示法	129	19.2.2 交互图和契约	165
17.9.1 类和实例的表示法	129		
17.9.2 链的表示法	130		
17.9.3 消息的表示法	130		
17.9.4 参数的表示法	131		

19.2.3	后置条件只是一个估计	166	20.1	引言	186
19.2.4	协作图和概念模型	166	20.2	对象之间的可见性	186
19.3	销售点系统的概念模型	166	20.3	可见性	187
19.4	销售点系统的协作图	167	20.3.1	属性可见性	187
19.5	协作图: <i>enterItem</i>	167	20.3.2	参数可见性	187
19.5.1	选择控制者类	168	20.3.3	局部声明可见性	188
19.5.2	显示商品描述信息和价格	169	20.3.4	全局可见性	190
19.5.3	创建一个新的 <i>Sale</i> 实例	169	20.4	可见性的 UML 表示法	190
19.5.4	创建一个新的 <i>SalesLineItem</i> 实例	170	第 21 章 设计类图		191
19.5.5	查找一个 <i>ProductSpecification</i>	171	21.1	引言	191
19.5.6	<i>ProductCatalog</i> 实例的可见性	171	21.2	活动及其相互之间的依赖关系	191
19.5.7	从一个数据库中检索 <i>ProductSpecification</i> 对象	171	21.3	何时创建设计类图	192
19.5.8	<i>enterItem</i> 的协作图	172	21.4	设计类图示例	192
19.5.9	发送给多对象的消息	172	21.5	设计类图	192
19.6	协作图: <i>endSale</i>	173	21.6	如何建立设计类图	193
19.6.1	选择控制者类	173	21.7	概念模型和设计类图的对比	194
19.6.2	设置 <i>Sale.isComplete</i> 属性值	173	21.8	建立销售点系统的设计类图	194
19.6.3	显示信息	174	21.8.1	识别出软件类并画出它们	194
19.6.4	计算销售项总额	174	21.8.2	添加方法名	195
19.6.5	计算销售项总额的协作图	175	21.8.3	方法名: 所要考虑的问题	196
19.7	协作图: <i>makePayment</i>	176	21.8.4	方法名—— <i>create</i>	196
19.7.1	选择控制者类	177	21.8.5	方法名——访问方法	196
19.7.2	创建支付项	177	21.8.6	方法名——多对象	197
19.7.3	记录销售项	178	21.8.7	方法名——独立于程序设计语言 的语法	197
19.7.4	计算余额	178	21.8.8	方法名——添加更多的 类型信息	197
19.8	协作图: <i>startUp</i>	180	21.8.9	添加关联和导航	198
19.8.1	何时建立 <i>startUp</i> 的协作图	180	21.8.10	添加依赖关系	201
19.8.2	应用程序如何启动	180	21.9	成员细节的表示法	202
19.8.3	<i>startUp</i> 系统操作的解释	181	21.10	样例模型	203
19.8.4	销售点应用系统的 <i>startUp</i> 操作	181	21.11	小结	203
19.8.5	选择初始化领域对象	182	第 22 章 系统设计要点		204
19.8.6	持久化对象: <i>ProductSpecification</i>	182	22.1	引言	204
19.8.7	<i>Store-create</i> () 的协作图	182	22.2	经典的三层体系结构	205
19.9	从表示层到领域层的连接	184	22.3	面向对象的多层体系结构	206
19.10	小结	185	22.3.1	应用逻辑层的分解	206
第 20 章 判定可见性		186	22.3.2	超过三层——多层体系结构	206
			22.3.3	部署	206
			22.3.4	多层体系结构的动机	207

22.4 用 UML 包描述体系结构	207
22.4.1 包的 UML 表示法	207
22.4.2 体系结构包图	208
22.4.3 包和依赖关系举例	208
22.5 包的识别	210
22.6 层和划分	210
22.7 两个包中类之间的可见性	211
22.8 服务包接口——虚包模式	212
22.9 窗口不直接对外可见——模型-视图 分离模式	212
22.9.1 模型-视图分离的意义	213
22.9.2 模型-视图分离和间接通信	214
22.10 一个系统中的间接通信	215
22.10.1 出版-订阅模式	215
22.10.2 回调	216
22.10.3 事件通知系统	216
22.11 应用协调者	217
22.11.1 应用协调者和窗口对象	218
22.11.2 自产的应用协调者	219
22.12 存储和持久化	219
22.13 样例模型	220

第五部分 构造阶段 (1)

第 23 章 设计到代码的映射	223
23.1 引言	223
23.2 程序设计与开发过程	223
23.2.1 构造阶段的创造性和改变	225
23.2.2 代码的改变和迭代的过程	225
23.2.3 代码的改变、CASE 工具和逆向 工程	226
23.3 将设计映射到代码	226
23.4 从设计类图创建类的定义	226
23.4.1 用方法和简单属性定义 一个类	226
23.4.2 加入引用属性	226
23.4.3 引用属性与角色名	228
23.5 根据协作图创建方法	228
23.5.1 POST 类的 <i>enterItem</i> 方法	228
23.5.2 POST 类的 <i>isNewSale</i> 方法	230
23.6 更新类的定义	231

23.7 代码中的容器/集合类	231
23.8 异常和错误处理	232
23.9 定义 <i>Sale</i> 类的 <i>makeLineItem</i> 方法	233
23.10 实现的顺序	233
23.11 从设计映射到代码的小结	234
第 24 章 用 Java 实现的程序方案	235

第六部分 分析阶段 (2)

第 25 章 选择第二个开发周期的 需求	243
25.1 第二个开发周期的需求	243
25.2 假定和简化	243
第 26 章 关联多个用况	245
26.1 引言	245
26.2 何时创建单独的用况	245
26.3 使用 <i>includes</i> 关系的用况图	245
26.4 使用 <i>includes</i> 关系的用况文档	246
26.4.1 例子	247
26.4.2 <i>Buy Items</i> 用况	247
26.4.3 <i>Pay by Cash</i> 用况	248
26.4.4 <i>Pay by Credit</i> 用况	248
26.4.5 <i>Pay by Check</i> 用况	249
第 27 章 扩展概念模型	250
27.1 销售点终端系统中的新概念	250
27.1.1 概念目录列表	250
27.1.2 针对用况采用名词短语策略来 查找概念	251
27.1.3 授权服务事务处理	252
27.1.4 POST 概念模型——草案 1	252
第 28 章 泛化	253
28.1 泛化	253
28.2 定义超类型和子类型	254
28.2.1 泛化和类型定义	254
28.2.2 泛化和类型集合	255
28.2.3 子类型定义的符合	255
28.2.4 子类型集合的符合	256
28.2.5 什么是正确的子类型	256
28.3 何时定义一个子类型	256
28.4 何时定义一个超类型	257
28.5 销售点终端系统的类型层次	258

28.5.1 支付类型	258	31.7 授权事务	281
28.5.2 授权服务类型	258	第 32 章 系统行为	283
28.5.3 授权事务类型	259	32.1 系统顺序图	283
28.6 抽象类型	261	32.1.1 <i>Buying Items</i> 用况的公共 起点	283
28.6.1 抽象类型的 UML 表示法	261	32.1.2 信用卡支付	283
28.6.2 抽象类和抽象方法	262	32.1.3 支票支付	283
28.7 对变化的状态建模	262	32.2 新的系统事件	283
28.8 类层次和继承	262	32.3 契约	284
第 29 章 包：组织模型元素的单位	264	第 33 章 状态图中的行为建模	287
29.1 导言	264	33.1 导言	287
29.2 包的 UML 表示法	264	33.2 事件、状态和转移	287
29.2.1 所有权和引用	264	33.3 状态图	288
29.2.2 包依赖	265	33.4 用况状态图	289
29.2.3 没有包图的包说明	265	33.5 系统状态图	290
29.3 如何划分概念模型	265	33.6 销售点应用系统的用况状态图	290
29.4 销售点终端系统的概念模型中 的包	266	33.6.1 <i>Buy Items</i> 用况	290
第 30 章 润饰概念模型	269	33.6.2 <i>Start Up</i> 用况	290
30.1 导言	269	33.7 需要状态图的类型	291
30.2 关联类型	269	33.7.1 状态无关和状态相关类型	291
30.3 聚合与组成	271	33.7.2 常见的状态相关类型和类	291
30.3.1 UML 中的聚合	271	33.8 销售点应用系统中的其他状态图	292
30.3.2 组成聚合——实心菱形	272	33.9 说明外部和内部事件	292
30.3.3 共享聚合——空心菱形	273	33.9.1 事件类型	292
30.3.4 如何确定聚合	273	33.9.2 内部事件的状态图	293
30.3.5 显示出聚合的好处	274	33.10 其他的状态图表示法	294
30.3.6 销售点终端模型中的聚合	274	33.10.1 转移动作和监护条件	294
30.4 关联角色的名称	274	33.10.2 嵌套状态	295
30.5 作为概念的角色与关联中的角色的 对比	275	第七部分 设计阶段 (2)	
30.6 派生元素	276	第 34 章 GRASP：用于职责分配的更多 模式	299
30.7 限定关联	277	34.1 GRASP：通用职责分配软件模式	299
30.8 递归关联或自反关联	278	34.2 多态	299
第 31 章 概念模型——总结	279	34.3 纯虚构	301
31.1 导言	279	34.4 中介者	303
31.2 领域概念包	279	34.5 “不要和陌生人讲话”	304
31.3 核心/混杂包	279	第 35 章 用更多的模式进行设计	308
31.4 支付	280	35.1 导言	308
31.5 产品	281	35.2 状态 (GoF)	308
31.6 销售	281		

35.3 多态 (GRASP)	312	37.5 用况驱动的开发	335
35.4 独身 (GoF)	314	37.6 系统结构上的早期重点	335
35.4.1 代码示例	316	37.7 开发中的各个阶段	336
35.4.2 独身的 UML 速记法	316	37.7.1 产品版本发布	336
35.4.3 实现	316	37.7.2 主要开发步骤	336
35.5 远程代理和代理 (GoF)	317	37.7.3 产品版本发布与开发周期	337
35.6 虚包和设备代理 (GoF)	319	37.7.4 计划和细化阶段	338
35.6.1 包装	319	37.7.5 以调查研究为重点的构造 阶段	339
35.6.2 虚包	319	37.7.6 主构造阶段	340
35.6.3 设备代理	320	37.7.7 实施阶段	340
35.6.4 中介者	320	37.7.8 开发周期中的分析阶段	340
35.6.5 封装和序列化	320	37.7.9 开发周期中的设计阶段	342
35.6.6 使用远程代理和设备代理	320	37.7.10 开发周期中的构造阶段	342
35.7 命令 (GoF)	321	37.7.11 开发周期中的测试阶段	342
35.7.1 <i>CreditPaymentApprovalReply</i> —— <i>execute</i> 消息	323	37.8 开发周期的长度	344
35.7.2 <i>CreditPaymentDenialReply</i> —— <i>execute</i> 消息	323	37.9 开发周期问题	345
35.8 结论	324	37.9.1 并行的开发小组和开发周期	345
		37.9.2 不明显的需求和技术体系 结构	345
		37.9.3 系统开发用况	346
		37.9.4 并行开发过程中的依赖关系	347
		37.10 体系结构层开发的时间调度	348
第八部分 专题		第 38 章 框架、模式和持久化	349
第 36 章 其他的 UML 表示法	327	38.1 引言	349
36.1 引言	327	38.2 问题: 持久化对象	349
36.2 通用的表示法	327	38.3 解决方案: 持久化框架	350
36.2.1 注解与约束	327	38.4 什么是框架	351
36.2.2 依赖关系	327	38.5 框架的功能	351
36.2.3 构造型与特性的规格说明	328	38.6 <i>PersistentObject</i> 超类	352
36.3 接口	328	38.7 关键思想	352
36.4 实现图	329	38.8 映射——用关系表来表示对象 模式	353
36.4.1 构件图	329	38.9 对象身份——对象标识符模式	353
36.4.2 实施图	329	38.10 代理——数据库代理模式	355
36.5 协作图中的异步消息	330	38.11 框架设计——模板方法模式	355
36.6 包接口	331	38.12 具体化——模板方法模式	356
第 37 章 开发过程问题	332	38.13 缓存对象——缓存管理模式	358
37.1 引言	332	38.14 智能引用——虚拟代理、桥模式	359
37.2 起因	332	38.14.1 泛化的虚拟代理	361
37.3 一个成功过程的指导原则	333		
37.4 迭代和增量开发	333		
37.4.1 瀑布模型的生命周期的缺点	333		
37.4.2 迭代开发的生命周期	333		

38.14.2 不需要公用的 <i>PersistentObject</i> 超类.....	361	38.18.3 被删除.....	369
38.14.3 虚拟代理的实现.....	363	38.18.4 提交操作.....	370
38.15 虚拟代理和数据库代理.....	363	38.18.5 回滚操作.....	370
38.15.1 虚拟代理和数据库代理的连续 ——工厂方法模式.....	363	38.19 在持久化存储器中寻找对象.....	371
38.15.2 为每件事物都设置一个代理.....	364	38.20 其他设计方案.....	372
38.16 如何用关系表来表示关系.....	365	38.20.1 元数据和参数化的代理.....	372
38.17 复杂对象实例化模式.....	365	38.20.2 查询对象.....	372
38.17.1 问题：具体化一个组合关系 层次.....	365	38.20.3 改变代理程序和内存中的数据 库代理.....	372
38.17.2 解决方案：请求式具体化.....	365	38.20.4 事务处理状态与多缓存.....	373
38.17.3 实例： <i>SalesLineItem</i> 的 具体化.....	366	38.21 尚待解决的问题.....	374
38.18 事务操作.....	368	附录 A 推荐读物	375
38.18.1 对象的事务状态.....	368	附录 B 样例开发活动和样例模型	377
38.18.2 变“脏”.....	369	参考文献	381
		术语表	383
		索引	387