<<面向对象分析与设计>>

图书基本信息

书名：<<面向对象分析与设计>>

13位ISBN编号：9787121173899

10位ISBN编号：7121173891

出版时间：2012-7

出版时间：电子工业出版社

作者：Grady Booch,Robert A.Maksimchuk,Michael W.Engle,Bobbi J.Young

页数：575

字数：850000

译者：王海鹏,潘加宇

前言

人类渴望得到精神上的宁静、美学上的成就、家庭的安全、正义和自由。
这一切都不能通过工业化的生产效率来直接满足。
但是，生产效率让人们得到充足的物质享受，而不至于与匮乏苦苦斗争。
这为精神、美学和家庭事务赢得了时间，也使得社会能够将一些特殊的技能赋予司法机构以及维护权利的机构。

Harlan Mills DPMA and Human Productivity 作为计算机专业人员，我们努力地去构建能工作而且有用的系统；作为软件工程师，我们面临着在计算资源和人力资源有限的条件下创建复杂系统的任务。
面向对象（OO）技术已经发展为管理许多不同种类的系统中内在复杂性的手段。
对象模型已被证明是非常有力和统一的概念。

对第2版的改动 在本书第2版出版以后，我们看到了一些重要的技术进步，其中一些突出的进步如下。

·与因特网的高带宽、无线连接已经非常普遍； ·纳米技术已经出现，并开始提供有价值的产品； ·机器人在火星表面漫步； ·计算机生成的特效使得在电影中能够完全逼真地再现任何想象中的世界； ·出现了个人气垫船； ·手机已无处不在，使用非常方便； ·获得了人类基因图谱； ·面向对象技术已经在工业软件开发中成为主流技术。

在世界各地都能见到面向对象技术被使用。
但是，我们仍然遇到许多人，他们还没有采用面向对象的开发方式。
对于这两类人，本书的新版本都很有价值。

对于面向对象分析与设计（OOAD）的新手，本书提供了下列信息。

·面向对象的概念支持和演进式的观点； ·如何在系统开发生命周期中应用OOAD的例子； ·对系统和软件开发中使用的标准表示法统一建模语言（UML 2.0）的介绍。

对于有经验的OOAD实践者，本书从不同的角度提供了价值。

·即使对于有经验的实践者，UML 2.0也是新的。
这是可以看到表示法方面的重要区别。

·根据前一版本所收到的反馈，更加关注建模。

·通过本书的概念部分的学习，可以了解在面向对象的世界中，“为什么事情总是像它们现在的样子”。
许多人可能从没研究过面向对象（OO）概念本身的发展，即使有所了解，在初次学习OO方法时，也许未能理解其重要性。

本书这一版和以前的版本相比有4项主要区别，如下所示。

1. UML 2.0已经正式得到了通过，第5章将介绍UML 2.0。
为了加强读者对这种表示法的理解，特别区分了它的基本元素和高级元素。

2. 这一版在应用程序的章节中引入了一些新的领域和背景。
例如，应用程序的领域范围很广，包括从高级系统架构到基于Web的系统的设计细节等各种不同层次的抽象。

3. 在前一版出版时，作为OO编程的概念来说，C++相对还是比较新的。
读者告诉我们，这种强调不再是主要的考虑。
现在有大量的OO编程和技术书籍及培训，还有许多为OO开发而设计的编程语言。
因此，大部分关于编码的讨论被删除了。

4. 最后，响应读者的要求，这一版更关注OOAD建模方面。
应用程序章节将展示如何利用UML，其中每一章强调了整个开发生命周期中的一个阶段。

本书的目标 本书在面向对象系统构建方面提供了实用指导。

它的具体目标如下。
·提供对对象模型的基础概念及其发展变化的正确理解；·帮助读者掌握面向对象分析和设计的表示法和过程；·介绍在不同的问题域中面向对象分析和设计的实际应用。
本书介绍的概念都基于牢固的理论基础，但本书首先是一本注重实效的书，面向架构师和软件开发者等软件工程实践者的实际需要。
读者对象　本书既是为计算机专业人员也是为学生编写的。
·对于实际系统和软件的开发者，本书将展示如何高效地利用面向对象技术来解决实际问题。
·对于系统分析师或架构师，本书将利用面向对象的分析与设计，提供一条从需求到实现的途径。
我们帮助分析人员或架构师提高识别能力，以区分不好的面向对象的结构与好的面向对象的结构，并在现实情况反常时权衡可选的设计方案。
也许最重要的就是，我们提供了一些让复杂系统变得有条理的新方法。
·对于项目经理，本书可以帮助他们更好地理解开发团队的资源分配、软件品质、测量指标以及管理与复杂软件系统相关的风险。
·对于学生，本书提供了一些必要的指导，使得学生能够开始掌握复杂系统开发的科学与艺术中的一些重要技巧。
本书不仅适合专业研讨班和个人学习使用，也适合作为高等院校本科生和研究生课程的教材。因为它主要阐述了软件开发的方法，所以非常适合软件工程和高级编程等课程，也可以作为涉及具体面向对象编程语言的课程的补充阅读材料。
本书的组织结构　本书分成3篇：概念、方法和应用，其中穿插了大量的补充材料。
概念　第1篇研究软件的内在复杂性及其表现方式。
本书将对象模型作为一种手段来帮助我们管理这种复杂性，详细地研究了对象模型的基本元素--抽象、封装、模块化、层次结构，讨论了“什么是类？
”以及“什么是对象？
”等基本问题。
由于确定有意义的类和对象是面向对象开发中的关键任务，因此我们花了相当多的时间来研究分类的本质。
具体来说，我们研究了生物学、语言学和心理学等其他学科中的分类方法，然后将这些经验应用到发现软件系统中类和对象的问题上。
方法　第2篇基于对象模型提出了复杂系统开发的一种方法。
针对面向对象的分析与设计，首先提出了一套图形表示法（即UML），然后是一个通用的过程框架。还研究了面向对象开发的实践，具体来说，就是它在软件开发生命周期中的位置以及它对于项目管理意味着什么。
应用程序　第3篇提供了一组（5个）不简单的例子，涉及不同问题域：系统架构、控制系统、密码分析、数据获取和Web开发。
之所以选择这些问题域，是因为它们是软件工程师实践过程中遇到的复杂问题的代表。
展示某些原则如何应用于简单的问题是很容易的，但是因为我们关注的是为现实世界构建有用的系统，所以我们对如何将对象模型应用于复杂应用程序更加感兴趣。
软件系统的开发不同于按菜谱做菜，因此我们强调应用程序的增量式开发，这种开发以一些正确的原则和良好的模型作为指导。
补充材料　本书中穿插了大量的补充资料。
多数章节中都有补充材料，这些材料对重要的主题提供了相关的信息。
本书包括了一个关于面向对象编程语言的附录，其中总结了一些常见语言的特征，还提供了常用术语的词汇表，以及一个扩展的分类参考书目，列出了关于对象模型的参考资料。
工具说明　读者总是会问创建本书中的图使用了什么工具。

我们主要使用两个很好的工具来画图：IBM Rational Software Architect和Sparx Systems Enterprise Architect。
为什么不只用一个？
市场的实际情况是，没有哪一种工具可以做所有的事情。
实践OOAD的时间越长，最后就会发现有些特别的情况是所有工具都不支持的。
（在这种情况下，可能需要寻求基本的绘图工具来展示你的想法。
）但是，不要让这些很少的情况阻止你使用健壮的OOAD工具，如我们提到的这两种工具。
本书的阅读方法　　对于本书可以一页一页地读，也可以不按现有的组织形式阅读。
如果想对对象模型中的基本概念或面向对象开发的动机有较深的理解，那么就应该从第1章开始依次读下去。
如果只对面向对象开发分析与设计中的表示法和过程感兴趣，就从第5章和第6章开始阅读。
第7章对使用这种方法管理项目的管理者来说特别有用。
如果对针对特定问题域的面向对象技术的应用程序更感兴趣，则可以在第8～12章中任选一章或者全部阅读。
致 谢　　我把本书献给我的妻子Jan，感谢她的爱和支持。
在第1版和第2版的写作过程中，一些人促成了我的面向对象开发思想。
对于他们的贡献，我特别要感谢，他们是：Sam Adams、Mike Akroid、Glenn Andert、Sid Bailin、Kent Beck、Dave Bernstein、Daniel Bobrow、Dick Bolz、Dave Bulman、Kayvan Carun、Dave Collins、Damian Conway、Steve Cook、Jim Coplien、Brad Cox、Ward Cunningham、Tom DeMarco、Mike Devlin、Richard Gabriel、William Genemaras、Adele Goldberg、Ian Graham、Tony Hoare、Jon Hopkins、Michael Jackson、Ralph Johnson、James Kempf、Norm Kerth、Jordan Kreindler、Doug Lea、Phil Levy、Barbara Liskov、Cliff Longman、James MacFarlane、Masoud Milani、Harlan Mills、Robert Murray、Steve Neis、Gene Ouye、Dave Parnas、Bill Riddel、Mary Beth Rosson、Kenny Rubin、Jim Rumbaugh、Kurt Schmucker、Ed Seidewitz、Dan Shiffman、Dave Stevenson、Bjarne Stroustrup、Dave Thomas、Mike Vilot、Tony Wasserman、Peter Wegner、Iseult White、John Williams、Lloyd Williams、Niklaus Wirth、Mario Wolczko和Ed Yourdon。
本书的相当一部分实践来自我参与并在世界各地开发的复杂软件系统，这些系统的开发公司包括Alcatel、Andersen Consulting、Apple、AT&T、Autotrol、Bell Northern Research、Boeing、Borland、Computer Sciences Corporation、Contel、Ericsson、Ferranti、General Electric、GTE、Holland Signaal、Hughes Aircraft Company、IBM、Lockheed、Martin Marietta、Motorola、NTT、Philips、Rockwell International、Shell Oil、Symantec、Taligent和TRW。
我曾有机会与数百名专业软件工程师和他们的经理协作，我要谢谢他们的帮助，是他们让本书与真实世界的问题相关。
特别要感谢Rational对我的工作的支持。
还要谢谢Tony Hall，他的卡通画给本书带来了亮点，否则这本书就只是一本乏味的技术书籍。
最后，我要谢谢我的3只猫--Cammy、Annie和Shadow，在我写作的许多个深夜，它们总是陪伴着我。
--Grady Booch　　我要感谢我的家人，他们必须忍受我参与编写这本书的漫长日子。
感谢我的父母，他们培育了我高尚的职业道德。
感谢Mary T.O' Brien，她为我提供了这个机会，这才使我开始了大量的后续工作。
感谢Chris Guzikowski帮助推动这项工作直至完成。
我要感谢合著者，感谢你们允许我加入这项工作，也感谢你们在这个项目中的努力工作和贡献。
最后，我要衷心感谢Grady多年前编写的本书的第1版，这本书是关于面向对象分析与设计最早的、最基础的书之一。
--Bob Maksimchuk　　我想表达对家人的感激，他们给了我爱和支持，这是我所有努力的基础。
感谢Grady给我机会，让我能够在他的经典著作的第3版中做出贡献。
最后，我要感谢Bob Maksimchuk在我成为一名作者的过程中所给予的指导。
--Mike Engle　　我要将本书献给我的母亲Jean Smith，她鼓励我参加这项工作。

我也要表达我对家人Russell、Alyssa和Logan的爱和感激，感谢他们的支持和鼓励。
感谢Bob Maksimchuk和Mike Engle，是他们让我有机会参与这项工作。
--Bobbi J. Young　　我要特别感谢我的丈夫Bob和两个孩子--Katherine和Ryan，他们的爱和支持给我了真正的灵感。
--Kelli A. Houston　　感谢我们的审稿者，特别是Davyd Norris和Brian Lyons。
感谢Addison-Wesley所有参与本书的其他工作人员，特别是Chris Zahn，他不仅参与了这项工作，而且保持了这项长时间工作的连贯性。

<<面向对象分析与设计>>

内容概要

本书是UML创始人Grady Booch的代表作之一。

全书分为理论和应用两部分。
理论部分深刻剖析了面向对象分析与设计（OOAD）的概念和方法。
应用部分连续列出了5个不同类型、不同领域的应用，描述如何从初始阶段到交付阶段，将OOAD理论和方法应用到项目中。
应用部分所涉及的领域包括系统架构、数据获取、密码分析、控制系统和Web开发，还给出了一些关于重要问题的有效建议，包括分类、实现策略和高性价比的项目管理。

书中介绍的概念都基于牢固的理论基础。
同时，作者十分注重实效，基于其丰富的经验，面向软件工程实践者的实际需要，提出了改进的对象开发方法，用于解决系统和软件开发者面临的复杂问题；运用大量例子说明了基本概念，解释了方法，并展示了在不同领域的成功应用。

书中的表示法采用最新的UML 2.0，因此本书是学习UML 2.0不可多得的参考书。

本书非常适合实际系统和软件的开发者、系统分析师或架构师、项目经理阅读，也可以作为高等院校软件工程和高级编程课程的教材使用。

作者简介

Grady Booch，在软件架构、软件工程和建模领域的创新工作是世界知名的。
从1981年Rational公司创建开始，他就一直担任该公司的首席科学家。
Grady于2003年3月成为了IBM院士（IBM Fellow）。

Grady是统一建模语言（UML）最早的开发者之一，也是几个Rational产品的最早开发者之一。
Grady曾担任世界各地一些复杂的软件密集型项目的架构师和架构指导者。

Grady是6本畅销书的作者，包括UML Users Guide和Object-Oriented Analysis with Applications。
Grady发表了几百篇有关软件工程的技术文章，其中包括在20世纪80年代早期发表的文章，这些文章最先提出了面向对象设计的术语和实践。
他曾在世界各地演讲和咨询。

Grady是美国计算机协会（ACM）、美国电气电子工程师学会（IEEE）、美国科学促进会（AAAS）、有社会责任的计算机专家协会（CPSR）的成员。
他是IBM院士、ACM院士、世界技术网络院士，也是软件开发论坛梦想家。
Grady是敏捷联盟、Hillside集团和软件架构师世界学院的创始委员会成员，也是Northface大学的顾问委员会成员。

Grady于1977年从美国空军学院获得学士学位，于1979年从加州大学圣巴巴拉分校获得电子工程科学硕士学位。

Grady与他的妻子和他的猫生活在科罗拉多。
他的兴趣包括阅读、旅行、唱歌和弹奏竖琴。

Robert A. Maksimchuk，是Unisys Chief Technology Office的一名研究主管。
他关注新出现的建模技术，目的是提升Unisys 3D可视企业建模框架的战略方向。
Bob为这项任务带来了不同行业的大量系统工程、建模、面向对象分析与设计的专业知识。
他是UML for Mere Mortals和UML for Database Design的合著者，也写了许多文章。
他曾经周游世界各地，在各种技术论坛上作为重要演讲者发言，举办关于UML和面向对象开发的研讨会和培训。
Bob是电气电子工程师学会（IEEE）和国际系统工程学会（INCOSE）的成员。

Michael W. Engle，是洛克希德马丁公司的首席工程师。
他有超过26年的技术和管理经验--从项目启动到运营支持，涵盖了完整的系统开发生命周期。
利用系统工程师、软件工程师和系统架构师的背景，Mike运用了面向对象技术，为复杂的系统开发提供创新的开发方式。

Bobbi J. Young， Ph.D.，是Unisys Chief Technology Office的一名研究主管。
她有着多年的IT行业从业经验，与商业公司和国防部合同供应商一同工作。
Young博士是一名咨询师，她在项目管理、企业架构、系统工程和面向对象分析与设计方面提供现场指导。
在她的职业生涯中，她关注于系统生命周期过程和方法学，同时也关注企业架构。
Young博士拥有生物学、计算机科学和人工智能学位，她获得了管理信息系统的博士学位，也曾是美国海军预备役的一名指挥官（已退伍）。

Jim Conallen，是IBM Rational的模型驱动开发战略小组的一名软件工程师。
在这个小组中，他积极参与，将对象管理集团（OMG）的模型驱动架构（MDA）计划应用于IBM Rational的模型工具中。
Jim在基于资产的开发和可复用资产规范（RAS）领域也很活跃。
Jim经常在会议上演讲，也经常写文章。
他的专业领域是Web应用开发。

他开发了UML的Web应用扩展（WAE）。
这是对UML的一种扩展，让开发者能够利用UML在合适的抽象和细节层面上对Web应用的架构进行建模。
这项工作是IBM Rational Rose和Rational XDE Web Modeling功能的基础。

Jim与人合著了两个版本的Building Web Applications with UML，第一个版本采用微软公司的ASP技术，后一个版本采用J2EE技术。

Jim的经验也来自于加入Rational之前的工作，那时他曾是独立的咨询师、Peace Corps的志愿者和大学讲师。
他还是3个孩子的父亲。
Jim从Widener大学获得了计算机和软件工程的学士学位和硕士学位。

Kelli Houston是IBM Rational的IT咨询专家。
她是IBM内部方法的方法架构师，负责编写方法并集成IBM的方法。
除了方法架构师的角色，Kelli还在IBM内部领导Rational Method Composer（RMC）特别兴趣小组（SIG）工作，为客户和IBM内部咨询师提供有效使用RMC方面的咨询和现场指导服务。

<<面向对象分析与设计>>

书籍目录

第1篇 概　念
第1章 复杂性
1.1 复杂系统的结构
1.1.1 个人计算机的结构
1.1.2 植物和动物的结构
1.1.3 物质的结构
1.1.4 社会机构的结构
1.2 软件固有的复杂性
1.2.1 定义软件复杂性
1.2.2 为什么软件在本质上是复杂的
1.3 复杂系统的5个属性
1.3.1 层次结构
1.3.2 相对本原
1.3.3 分离关注
1.3.4 共同模式
1.3.5 稳定的中间形式
1.4 有组织和无组织的复杂性
1.4.1 复杂系统的规范形式
1.4.2 人在处理复杂性时的能力局限
1.5 从混沌到有序
1.5.1 分解的作用
1.5.2 抽象的作用
1.5.3 层次结构的作用
1.6 复杂系统的设计
1.6.1 作为科学和艺术的工程
1.6.2 设计的含义
1.7 小结
第2章 对象模型
2.1 对象模型的演进
2.1.1 程序设计语言的换代
2.1.2 第一代和第二代早期程序设计语言的拓扑结构
2.1.3 第二代后期和第三代早期程序设计语言的结构
2.1.4 第三代后期程序设计语言的结构
2.1.5 基于对象和面向对象的程序设计语言的结构
2.2 对象模型基础
2.2.1 面向对象编程
2.2.2 面向对象设计
2.2.3 面向对象分析
2.3 对象模型要素
2.3.1 抽象的意义
2.3.2 封装的意义
2.3.3 模块化的意义
2.3.4 层次结构的意义
2.3.5 类型的意义
2.3.6 并发的意义

2.3.7 持久的意义
2.4 应用对象模型
2.4.1 对象模型的好处
2.4.2 开放式问题
2.5 小结
第3章 类与对象
3.1 对象的本质
3.1.1 什么是对象，什么不是对象
3.1.2 状态
3.1.3 行为
3.1.4 标识符
3.2 对象之间的关系
3.2.1 链接
3.2.2 聚合
3.3 类的本质
3.3.1 什么是类，什么不是类
3.3.2 接口和实现
3.3.3 类的生命周期
3.4 类之间的关系
3.4.1 关联
3.4.2 继承
3.4.3 聚合
3.4.4 依赖关系
3.5 类与对象的互动
3.5.1 类与对象的关系
3.5.2 类与对象在分析和设计中的角色
3.6 创建高品质的类与对象
3.6.1 评判一种抽象的品质
3.6.2 选择操作
3.6.3 选择关系
3.6.4 选择实现
3.7 小结
第4章 分类
4.1 正确分类的重要性
4.1.1 分类的困难
4.1.2 分类的增量和迭代本质
4.2 确定类和对象
4.2.1 经典方法和现代方法
4.2.2 面向对象分析
4.3 关键抽象与机制
4.3.1 确定关键抽象
4.3.2 识别机制
4.4 小结
第2篇 方　法
第5章 表示法
5.1 统一建模语言
5.1.1 简单历史回顾

5.1.2 模型与多重视图
5.1.3 图分类
5.1.4 在实践中使用图
5.1.5 概念模型、逻辑模型和物理模型
5.1.6 工具的角色
5.1.7 面向对象开发的产品
5.1.8 规模上的伸缩
5.1.9 UML的语法和语义
5.1.10 UML 2.0信息资源
5.2 包图
5.2.1 基本概念：包表示法
5.2.2 基本概念：元素的可见性
5.2.3 基本概念：依赖关系
5.2.4 基本概念：包图
5.2.5 高级概念：导入和访问
5.3 组件图
5.3.1 基本概念：组件表示法
5.3.2 基本概念：组件图
5.3.3 基本概念：组件接口
5.3.4 基本概念：组件实现
5.3.5 高级概念：组件的内部结构
5.4 部署图
5.4.1 基本概念：工件表示法
5.4.2 基本概念：节点表示法
5.4.3 基本概念：部署图
5.5 用例图
5.5.1 基本概念：执行者
5.5.2 基本概念：用例
5.5.3 基本概念：用例图
5.5.4 高级概念：“ include ”和“ extend ”关系
5.5.5 高级概念：泛化
5.6 活动图
5.6.1 基本概念：动作
5.6.2 基本概念：开始和停止
5.6.3 基本概念：判断节点和合并节点
5.6.4 基本概念：分区
5.6.5 高级概念：分叉、结合和并发
5.6.6 高级概念：对象流
5.6.7 高级概念：其他元素
5.7 类图
5.7.1 基本概念：类表示法
5.7.2 基本概念：类关系
5.7.3 高级概念：模板（参数化）类
5.7.4 高级概念：可见性
5.7.5 高级概念：关联端名称和限定符
5.7.6 高级概念：约束
5.7.7 高级概念：关联类和注解

5.8 序列图
5.8.1 基本概念：对象与交互
5.8.2 基本概念：生命线与消息
5.8.3 高级概念：销毁事件
5.8.4 高级概念：执行说明
5.8.5 高级概念：交互使用
5.8.6 高级概念：控制结构
5.9 交互概述图
5.9.1 基本概念：框
5.9.2 基本概念：控制流元素
5.9.3 基本概念：交互图元素
5.10 组合结构图
5.10.1 基本概念：组合结构的部分
5.10.2 基本概念：组合结构的部分与接口
5.10.3 基本概念：组合结构连接器
5.10.4 高级概念：协作
5.11 状态机图
5.11.1 基本概念：初始状态、最终状态和简单状态
5.11.2 基本概念：转换与事件
5.11.3 高级概念：状态活动--入口活动、执行活动和出口活动
5.11.4 高级概念：控制转换
5.11.5 高级概念：复合状态与嵌套状态
5.11.6 高级概念：并发与控制
5.11.7 高级概念：子状态机状态
5.11.8 高级概念：其他状态机图元素
5.12 时间图
5.12.1 基本概念：更多相同之处
5.12.2 基本概念：布局
5.12.3 基本概念：事件
5.12.4 基本概念：约束
5.12.5 高级概念：另一种表示形式
5.12.6 高级概念：事件与消息
5.13 对象图
5.13.1 基本概念：对象
5.13.2 基本概念：对象关系
5.13.3 高级概念：端点名称和限定符
5.14 通信图
5.14.1 基本概念：对象、链接和消息
5.14.2 基本概念：顺序表达式
5.14.3 高级概念：消息与同步
5.14.4 高级概念：迭代子句和警戒条件
5.15 小结
第6章 过程
6.1 首要原则
6.1.1 成功项目的特征
6.1.2 追求理性的开发过程
6.2 宏观过程：软件开发生命周期

6.2.1 概述
6.2.2 宏观过程的内容维：科目
6.2.3 宏观过程的时间维：里程碑和阶段
6.2.4 宏观过程的时间维：迭代
6.2.5 发行计划
6.3 微观过程：分析与设计过程
6.3.1 概述
6.3.2 抽象层次
6.3.3 活动
6.3.4 产品
6.3.5 微观过程与抽象层次
6.3.6 识别元素
6.3.7 确定元素间的协作
6.3.8 确定元素间的关系
6.3.9 详细确定元素的语义
6.4 小结
第7章 实战
7.1 管理和计划
7.1.1 风险管理
7.1.2 任务计划
7.1.3 开发复查
7.2 人员配备
7.2.1 资源配置
7.2.2 开发团队角色
7.3 发布版本管理
7.3.1 配置管理和版本控制
7.3.2 集成
7.3.3 测试
7.4 复用
7.4.1 复用的元素
7.4.2 建立复用制度
7.5 质量保证和测量指标
7.5.1 软件质量
7.5.2 面向对象测量指标
7.6 文档化
7.6.1 开发遗产
7.6.2 文档化的内容
7.7 工具
7.7.1 工具种类
7.7.2 组织上的意义
7.8 特殊主题
7.8.1 领域特定问题
7.8.2 采纳面向对象技术
7.9 面向对象开发的好处和风险
7.9.1 面向对象开发的好处
7.9.2 面向对象开发的风险
7.10 小结

第3篇 应　用
第8章 系统架构--基于卫星的导航
8.1 初始
8.1.1 卫星导航系统的需求
8.1.2 定义问题的边界
8.1.3 确定任务用例
8.1.4 确定系统用例
8.2 细化
8.2.1 开发一个好的架构
8.2.2 定义架构开发活动
8.2.3 验证所建议的系统架构
8.2.4 分配非功能需求和确定接口
8.2.5 规定系统架构及其部署
8.2.6 分解系统架构
8.3 构造
8.4 交付之后
8.4.1 添加新的功能
8.4.2 改变目标硬件
第9章 控制系统--交通管理
9.1 初始
9.1.1 列车交通管理系统的需求
9.1.2 决定系统用例
9.2 细化
9.2.1 分析系统功能
9.2.2 定义TTMS架构
9.2.3 从系统工程到硬件和软件工程
9.2.4 关键抽象和机制
9.3 构造
9.3.1 消息传送
9.3.2 列车时刻表计划
9.3.3 显示信息
9.3.4 传感器数据采集
9.3.5 发布版本管理
9.3.6 系统架构
9.3.7 子系统规格说明
9.4 交付之后
第10章 人工智能--密码分析
10.1 初始
10.1.1 密码分析需求
10.1.2 定义问题的边界
10.1.3 黑板框架的架构
10.1.4 知识源的分析
10.2 细化
10.2.1 黑板对象
10.2.2 依赖和认定
10.3 构造
10.3.1 设计黑板对象

10.3.2 设计知识源
10.3.3 设计控制器
10.3.4 集成黑板框架
10.3.5 添加新的知识源
10.4 交付之后
10.4.1 系统增强
10.4.2 改变需求
第11章 数据采集--气象监测站
11.1 初始
11.1.1 气象监测站需求
11.1.2 定义问题的边界
11.1.3 场景
11.2 细化
11.2.1 气象监测系统用例
11.2.2 架构框架
11.3 构造
11.3.1 帧机制
11.3.2 发布计划
11.3.3 传感器机制
11.3.4 显示机制
11.3.5 用户界面机制
11.4 交付之后
第12章 Web应用--休假跟踪系统
12.1 初始
12.1.1 需求
12.1.2 用例模型
12.2 细化
12.2.1 部署视图
12.2.2 逻辑视图
12.2.3 进程视图
12.2.4 实现视图
12.2.5 用例视图
12.3 构造
12.3.1 用户体验模型
12.3.2 分析和设计模型
12.3.3 实体
12.3.4 控制器
12.3.5 Web页面和用户界面
12.4 交付和交付之后
附录A 面向对象编程语言
附录B 进一步阅读
注解
术语表
分类书目

章节摘录

版权页： 插图： 2.时间和空间语义 当确定存在某个操作，并定义了它的功能语言之后，必须决定它的时间语义和空间语义。
这意味着我们必须决定它完成操作需要的时间以及存储空间。
这样的决定通常是用最佳、平均和最差等术语来表达的，最差的情况规定了能够接受的上限。
前面也曾提到，当一个对象通过一个链接向另一个对象传递消息时，这两个对象必须以某种方式同步。
在多控制线程的情况下，这意味着消息的传递比子程序调用要更复杂。
在使用的大多数语言中，对象之间的同步不成问题，因为我们的程序只包含一个控制线程，这意味着所有的对象都是被依次访问的。
这种情况下的消息传递很简单，因为它的语义基本上与简单的子程序调用相同。
但是，在支持并发的语言中，我们必须关注更为复杂的消息传递形式，以避免两个控制线程以无限制的方式访问同一个对象，从而引发问题。
前面曾提到，在多个控制线程下仍能保持语义的对象要么是守卫的，要么是同步的。
3.6.3选择关系 在类之间和对象之间选择关系与选择操作是有联系的。
如果决定对象x向对象Y发送消息M，那么X必须能够直接或间接地访问Y；否则，就不能够在X的实现中命名操作M。
所谓能够访问，指的是一种抽象能够看到另一种抽象，并引用它的外部视图中的资源。
只有当它们的范围重叠，并且访问得到授权时（例如，类的私有部分只能被该类本身和它的友元访问），一种抽象才可以访问另一种抽象。
因此，耦合是度量可访问程度的指标。
1.Demeter法则 在选择对象间的关系时，有一条有用的指导原则，称为Demeter法则。
它指出，“类的方法不应该以任何方式依赖于任何类的结构，除了它自己类的当前（顶层）结构之外。
而且，每个方法只能够对一个非常有限的类集的对象发出消息”[56]。
应用这一法则的基本效果就是创建了一些松耦合的类，它们的实现秘密被封装了起来。
这样的类是相当没有负担的，即为了理解一个类的意思，不需要理解许多其他类的细节。

编辑推荐

书中的表示法采用最新的UML2.0，因此《面向对象分析与设计(第3版)》是学习UML2.0不可多得的参考书。
《面向对象分析与设计(第3版)》非常适合实际系统和软件的开发者、系统分析师或架构师、项目经理阅读，也可以作为高等院校软件工程和高级编程课程的教材使用。
原著中所附“分类书目”可从http://www.broadview.com.cn/17389下载。

<<面向对象分析与设计>>

版权说明