世界著名计算机教材精选

PEARSON

面向对象软件工程

使用UML、模式与Java（第3版）

Bernd Bruegge
Allen H. Dutoit 著
叶俊民 汪望珠 等译

OBJECT-ORIENTED SOFTWARE ENGINEERING

Using UML, Patterns, and Java, Third Edition

清华大学出版社

PEARSON Education

世界著名计算机教材精选

面向对象软件工程

使用 UML、模式与 Java

（第 3 版）

Bernd Bruegge
Allen H. Dutoit 著
叶俊民　汪望珠　等译

清 华 大 学 出 版 社
北　京

EISBN:978-0-13-606125-0

图书在版编目（CIP）数据

面向对象软件工程：使用 UML、模式与 Java(第 3 版)/(美)布吕格(Bruegge, B.),(美)迪图瓦(Dutoit. A. H.)著；叶俊民，汪望珠等译. —北京：清华大学出版社，2011.1

书名原文：Object-Oriented Software Engineering Using UML, Patterns, and Java, 3e

(世界著名计算机教材精选)

ISBN 978-7-302-23814-0

Ⅰ. ①面…　Ⅱ. ①布… ②迪… ③叶… ④汪…　Ⅲ. ①面向对象语言－软件工程－教材 ②JAVA 语言－程序设计－教材　Ⅳ. ①TP311.5 ②TP312

中国版本图书馆 CIP 数据核字(2010)第 174977 号

责任编辑：龙啟铭

责任校对：李建庄

责任印制：何　芊

出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座

http://www.tup.com.cn　　**邮　　编**：100084

社　总　机：010-62770175　　**邮　购**：010-62786544

投稿与读者服务：010-62795954, jsjjc@tup.tsinghua.edu.cn

质　量　反　馈：010-62772015, zhiliang@tup.tsinghua.edu.cn

印 刷 者：北京密云胶印厂

装 订 者：北京市密云县京文制本装订厂

经　　销：全国新华书店

开　　本：185×260　　**印　张**：37.25　　**字　数**：924 千字

版　　次：2011 年 1 月第 1 版　　**印　次**：2011 年 1 月第 1 次印刷

印　　数：1～3000

定　　价：69.50 元

产品编号：035082-01

译者的话

2009年下半年的某一天，我接到清华大学出版社龙启铭编辑的电话，龙编辑在电话中告知我：由卡耐基·梅隆大学（CMU）教授Bernd Bruegge等人撰写的《面向对象软件工程》（第3版）英文版刚刚出来，问我是否能够将该书翻译一下。拿到样书后，我发现第3版除了修订第2版中出现的问题外，主要完善和增加了如下内容：

（1）全面使用最新的UML以及OCL标准。作者修改了本书中大部分的图表，以利用UML和OCL最新的进展。特别是作者系统和对象设计中使用了有球窝记号的构件图。

（2）扩展了敏捷方法。在第2版中，作者在第16章引入了XP技术。在这一版本中，作者把敏捷材料扩展到Scrum和Rugby，并在第11、13和14章中进行了更新，以用于测试、软件配置管理和项目管理方面。

（3）增加了连续集成的新内容。使用敏捷方法的一种做法是不断地把软件变化集成为主要的产品主线上。尽管这一做法能够找到并解决集成问题，这一方法在实现时也会遇到很多挑战。作者在第13章“软件配置管理”中提出了这一新内容。

（4）增加了关于U2TP和自动测试的新内容。在作者的教学中他们发现UML2测试概况的扩展有助于讨论测试概念，特别是有助于区分测试系统和待测系统。这也使得作者得以把测试材料扩展到自动测试和自动化测试生成上。

（5）改进了贯穿始终的案例研究和实例。

在接下任务后，我们迅速组织了工作团队，并开始了艰苦和细致的翻译工作。具体分工是这样的：第1～5，15章的初译稿由叶俊民和汪望珠完成；第6～7章的初译稿由朱凯完成；第8～9章的初译稿由张涛完成；第10，16章的初译稿由熊华根完成；第11章的初译稿由彭波涛和叶俊民完成；第12～13章的初译稿由郑艳艳完成；第14章的初译稿由李蓉完成；术语表的初译稿由朱凯、叶俊民和冯星星完成；其他部分的初译稿，如前言、序言、致谢、封底和附录A等，由汪望珠完成。在初译稿的基础上，叶俊民和汪望珠完成了终译和统稿。研究生冯星星、谢茜、李明、綦志勇、肖光、郑全、钱茅、朱高华等人参与了文字校对和插图绘制和处理工作，在此过程中，他们发现了并改正了很多错误。在此，作为主译者，我们希望对所有参加本次翻译工作的教师和学生表示衷心感谢！感谢大家的辛勤劳动和任劳任怨，这为本书翻译的顺利完成打下了基础。

现在，关于软件工程的教材或者教学参考书很多，各个本子均有其优点和长处，均能够适应某一读者群。实事求是讲，B.Bruegge等人撰写的《面向对象软件工程》是一本非常棒的教材和教学参考书，非常适合于作为我国高校高年级本科生和低年级研究生的教材或教学参考书，书中的实例非常多，且无论是概念还是过程，该书均讲授得很细致，按照书中的指示和建议去学习做项目，学生们会发现原来做项目是很容易入手和深入的。此外，我本人从翻译过程中学习了很多知识和技术，我将从中汲取到的很多“营养”运用于我的教学和写作中，因此我想借此序表达一下对该书作者的敬意和感谢。因此，我本人是非常

愿意将这一好书推荐给广大读者。当然一本好的翻译教材是否能够得到广大教师、学生和科技人员的肯定，取决于很多因素，比如教材本身、翻译质量，等等。

最后，译者所在团队虽然非常努力工作并希望将这一好书译好并奉献给广大读者，但因水平有限，难免在翻译中有词不达意之处，敬请广大读者批评指正！

前　言

十多年以前，我就了解到由卡耐基·梅隆大学（CMU）的 Bernd Bruegge 讲授的一门软件工程课程。在很多其他大学的软件工程课程上，通常的做法是，在一个学期中，将 3～4 个学生分在一个小组中，并给每一个小组分派几个小问题或项目，其中每一个小问题或项目的研究周期不超过一个月。在这些小项目中，有一个能力较强的主程序员，通常通过该主程序员的强力带动而推动整个小组的工作，以完成这些小项目。在这样的背景之下，通常学生没有必要学习沟通技能，没有必要使用建模工具，也没必要处理实际问题中所存在的歧义性。在这种环境下培养的学生并没有学会怎样去处理开发实际项目时将遇到的各种复杂性问题。在 Bruegge 教授的课程中，全班同学在整个学期中都在完成同一个项目：为匹兹堡市开发一个面向查询的导航系统。学生们得在上一个学期学生所开发出的交互系统的基础上，完成进一步的开发。客户是市规划部门的经理和领域权威。在该项目中，地理数据和汽车调度数据存在不精确、格式不兼容的情况，在学年结束时，学生们完成了一个超过 27 000 行代码的系统，该系统最终被客户接受。这一结果与许多其他院校教学中使用的软件工程小项目相比，有多么大的不同啊！CMU 的学生通过学习该课程，理解到了处理现实世界复杂性和杂乱性所需要的策略、组织和工具。学生们通过在实例中实践来学习软件工程课程，这是学会任何技艺的必由之路。

本书反映的是将软件开发作为一门工程学科的实用哲学。作者采用一种观点——一种使用 UML 面向对象的方法，这使得软件工程的许多方面能够为学生所了解到。这些内容包括完成实际项目所需要的建模技术、人与人之间的沟通技巧。除此之外，还包含了如何管理变化的章节内容，这是每一个实际项目中均会出现的话题，但这一内容在其他的软件工程书籍中又常常被忽略掉。阅读本书将会使读者对软件工程的丰富范畴及其复杂性有深刻的理解。

我特别欣赏本书中的那些从广泛领域中选取出来的能够给人以启示的轶事。这些轶事提供或大或小的活生生的问题实例，阐述了工程师们必须面对的微妙问题和陷阱。任何阐述玻利尼西亚人航行的相关实例、那一种盘根错节的 Tolkien《指环王》的历史以及祖母切火腿的秘方的书籍等实例不仅实用，而且读起来也妙趣横生。

Jim Rumbaugh

序　言

高达 8611 米的 K2 山峰高高地耸立在西喜马拉雅的喀喇昆仑山脉，它是世界上第二高的山峰，被认为是高度在 8000 米以上最难攀登的山峰。对 K2 山峰的探险，即便是在天气最适宜的夏季，通常也要用连续几个月左右的时间。而且，即使在夏季，暴风雪的天气也是很频繁出现的。进行一次登山探险活动，通常需要数千磅的设备，这些设备包括攀登齿链、抵御恶劣天气的防护装备、帐篷、食物、通信设备和支付给数以百计的搬运工人的薪水和鞋子。策划这样一次探险性登山活动，需要花费一个登山者一生大量的时间，还需要数十个后援者协助。一旦开始攀登，会存在很多难以预料的突发事件，如雪崩、搬运工人罢工和设备故障等，这将迫使登山者调整原有计划、找出解决问题的新办法或者决定回撤。目前，对 K2 峰的探险成功率不到 40%。

美国国家空间系统（NAS）监管并控制着美国的空中交通。NAS 拥有 18 300 多个航空站点，21 个空中交通控制中心和 460 余个控制塔。这些站点和中心拥有的设备总数超过 34 000 件，这些设备包括雷达系统、信息交换机、无线电收发设备、计算机系统和显示设备等。当前的情况是这些基础设施老化得很快。例如，支持 21 个空中交通控制中心中的计算机系统是 20 世纪 80 年代早期购买的 IBM 3083 大型机。从 1996 年起，美国政府启动了一个项目来更新 NAS 基础设施，以使之现代化，这一更新计划包括改善卫星导航系统、改善数字控制器以及飞行员通信系统，并且在空中路线控制、飞机着陆顺序安排、飞机驶离和驶向跑道时所涉及的地面交通控制等方面实现更高程度的自动化。但是，想让如此复杂系统的基础设施现代化，只能循序渐进地工作。因此，当引入具有新功能的设备时，仍要支持原有设备的正常运转。例如，在这一过渡时期内，控制器将不得不同时具有模拟和数字两种声音通道，以实现与飞行员的通信。最后，NAS 的现代化和全球空中交通同时发生巨大增长，后者预计在未来 10～15 年后，会成倍增长。被称为“先进自动化系统（AAS）”的 NAS 的最初现代化尝试，因存在与软件相关的问题，在开始延误几年以及在花费上超出预算数十亿美元后于 1994 年被迫取消。

上面的两个实例所讨论的均是复杂系统，在此，外部条件将引发意想不到的变化。问题的复杂性超出了任何人可以控制的范畴。变化使得参与者放弃了常用的方法而创造新的解决方案。在上述两个实例中，有多名参与者要求进行合作，以开发出新的技术来对付这些挑战。因为，如果不这样做，我们将无法达到目标。

本书就是关于如何处理复杂且不断变化的软件系统这一主题的。

主题

应用域（登山计划、空中交通控制、金融系统以及文字处理等）通常包括很多软件开发者不熟悉的概念。而解答域（用户界面工具包、无线通信、中间件、数据库管理系统、交易处理系统和嵌入式计算机系统等）则通常是不成熟的，而且会给开发者带来很多具有挑战性的实现技术。因此，系统和开发项目都是复杂的，涉及很多不同的构件、工具、方

法和人员。

当开发者对用户的应用域有了更多的理解之后，用户们将会更新系统需求。当开发者对正在出现的技术或者对现行技术的局限性有了更多了解之后，他们会调整系统的设计和实现。当质量控制发现系统中的缺陷，以及用户要求增加新功能时，开发者将修改系统及其相关的工作产品，这一切都将导致不断的变化。

复杂性和不断发生的变化，代表了这样一些挑战：变化使得任何个人都不可能去控制系统及其演化。如果控制不当，复杂性和不断的变化也将使得解决方案在宣布之前就流产。如果对应用域的理解出现错误，会使得出台的解决方案对用户毫无用途，从而造成不得不返工的情形。不成熟或不兼容的实现技术，会造成系统的稳定性变差，以及系统被延迟交付。如果处理不了变化，将带来新的系统缺陷，从而使得系统性能降低到无法使用的程度。

本书反映了作者 10 多年来构造系统以及教授软件工程课程的体会。我们发现，学生常常孤立地学习程序设计技术和软件工程技术，常常选择小的问题作为研究实例。所带来的结果是，学生们能够有效地解决定义明确的问题，但当他们第一次真正面对复杂而真实的开发项目时，常常会感到束手无策。真实的开发过程需要很多不同的技术和工具，要求开发人员之间进行合作。针对这一情况，现今的软件工程本科课程体系中通常包括一门软件工程项目管理的课程，讲授组织成一个开发项目所必需的知识。

工具：UML、Java 和设计模式

编写本书时，在我们的头脑中一直有一个课程项目。这个项目除了可以作为课程设计的项目之外，还可以在其他场合使用，例如用作短期集训或者短期研发的项目等。我们使用了真实系统中的实例，并检验 UML、基于 Java 的技术、设计模式、设计原理、配置管理以及质量控制等现代技术之间的交互。此外，我们讨论了与项目管理相关的问题，这些问题与上述技术及其对复杂性和变化的影响相关。

原则

我们遵循如下 5 条原则来进行软件工程教学：

实践经验 我们认为软件工程教学必须与实践结合起来。学生只有参与开发一个复杂系统（即一个没有任何学生能够完全理解的系统），才能理解什么是复杂性。

问题解决 在我们看来，软件工程教学必须建立在解决实际问题基础之上。因此，没有绝对正确或错误的解决方案，只有文档标准相对做得比较好的解决方案和做得较差的解决方案。尽管我们检验了现有的解决方案并鼓励重用这些方法，但同时我们也鼓励对标准解决方案的批评与改进。

有限资源 如果拥有充足的时间和资源，我们将能够建立起理想的系统。这样的系统建立存在着一些问题。首先，这一想法是不现实的，其次，即使在开发过程中具有足够的资源，如果最初要解决的问题在开发过程中很快发生了变化，那么我们最终将交付一个解决了错误问题的系统。因此，我们有必要假设问题的解决过程在资源方面受到限制。同时，我们强烈地意识到资源的缺乏将促使开发者使用基于构件的开发方法、重用知识、设计方案和编码。换句话讲，我们将支持用工程方法进行软件开发。

跨学科性 软件工程是一个跨学科的领域。软件工程要用到电子工程、计算机工程、计算机科学、商务管理、图形设计、工业设计、体系结构、戏剧与写作等领域的知识。软

件工程是一个应用领域。当开发者试图理解并对应用域进行建模时，开发者与其他人员包括用户和客户进行定期的沟通，一些用户和客户可能对软件开发知之甚少。这就要求从多视角、使用多种术语来看待和处理系统。

沟通　即使是开发者仅仅为开发者建构软件，他们仍然需要在他们自己之间进行沟通。作为开发者，我们不能够只限于与同行进行沟通。我们还需要讨论被选方案、阐述解决方案、协商交易，以及评审和评价其他人的工作。大量失败的软件工程项目都可追溯到信息表达不准确，以及信息的遗失。我们必须学会和所有的项目参与者进行沟通，这其中最重要的参与者就是客户和最终用户。

上述 5 条原则是本书的基础。这些原则将促进并让读者能够使用符合实际的、现代的解决方案来解决复杂和不断变化的问题。

关于本书

本书基于应用于软件工程的面向对象技术。本书既不是一本探讨所有可能方法的软件工程概论，也不是一本关于算法和数据结构的程序设计教程。相反地，我们将重点放在有限的一些技术上，并且解释这些技术在适度复杂环境中的应用，例如一个包含 20～60 个参与者参加的多组开发项目。因此，本书也反映了我们本身的偏好、实力和弱点。尽管如此，我们希望所有的读者都能够从本书中找到他们所需要的东西。本书共有 16 章，分为 3 个部分，可作为一个学期的教学课程。

第 1 部分“开始”包含 3 章。在这一部分中，重点介绍了软件工程环境中的开发者所必须具备的基本技能。

- 第 1 章“软件工程导论”描述了程序设计和软件工程之间的差异、本学科当前存在的挑战，以及全书中用到的概念的基本定义。
- 第 2 章“使用 UML 进行建模”描述了面向对象技术中所用到的建模语言 UML 的基本组成。我们将建模作为处理复杂性的一项技术提出来。本章教授读者如何阅读和理解 UML 图。在随后的章节中，我们将讲解如何创建 UML 图来对系统不同方面进行建模。在全书中，我们使用 UML 为多种不同的产品进行建模：从软件系统到软件过程和工作产品等。
- 第 3 章“项目组织和沟通”，介绍了一些项目组织和沟通的基本概念。开发者和管理者将一半以上的时间花在与其他人沟通上，所使用的沟通方法主要是面对面进行沟通，以及通过电子邮件、群件、电视会议或书面文件等方式进行沟通。建模用于处理复杂性，沟通则用于处理变化。我们描述项目组织以及探讨如何才能进行有效的沟通。

第 2 部分“复杂性处理”集中探讨那些有助于开发者详细说明、设计和实现复杂系统的技术和方法。

- 第 4 章“需求获取”和第 5 章“分析”两章从用户角度描述了系统定义。在需求获取的过程中，开发者决定用户所需求的功能，以及达到这一功能的可行方法。在分析阶段，开发者将上述理解形式化，并保证其完整性和一致性。在此，我们主要讨论如何使用 UML 来处理应用域的复杂性。
- 第 6 章 “系统设计：分解系统”和第 7 章“系统设计：选择设计目标”两章从开

发者角度出发，对系统进行了定义。在这一阶段，开发者使用了设计目标和子系统分解的形式，定义了系统的结构。开发者探讨了全局性问题，如：系统到硬件的映射、持久数据的存储以及全局控制流等。讨论集中在开发者如何使用体系结构风格、构件以及 UML 来处理解答域的复杂性。

- 第 8 章“对象设计：复用模式解决方法”、第 9 章“对象设计：说明接口”和第 10 章“将模型映射到代码”描述了与解答域相关的具体建模活动和构建活动。在这一阶段，开发者找出并调整设计模型和框架以实现某些具体子系统。他们使用诸如 UML 对象约束语言（Object Constraint Language）的约束语言来精化和精确说明类接口。最后，他们将详细的对象设计模型映射到源代码和数据库框架上。
- 第 11 章“测试”描述了对照系统模型对系统行为的验证。通过测试可以发现系统中的错误，包括那些系统发生变化以及系统需求发生变化时产生的错误。测试活动包括单元测试、集成测试和系统测试。我们将介绍几种测试技术，包括：白盒测试、黑盒测试、路径测试、基于状态的测试和检测，并讨论这些测试技术在面向对象系统中的应用。

第 3 部分“变更管理”集中讨论了在整个系统开发中支持变化的控制、评价以及实现的方法和技术。

- 第 12 章“基本原理管理”描述了设计决策的获取，以及对这些设计决策的论证。我们在需求获取、分析和系统设计这三个阶段所开发出来的模型，对一个系统应该做什么和应该如何去做，提供了不同的视点，这将有助于复杂问题的处理。为了能够处理变化，我们也需要知道系统为什么是这样的。获取设计决策、可选方案的评估以及对这些方案的论证，使我们得以了解系统的基本原理。
- 第 13 章“配置管理”描述了对项目开发过程进行建模的技术和工具。配置管理有助于我们对变化进行处理，这样就补充了基本原理。版本管理记录了系统进化。发行管理保证了一次发行的各个构件的质量，以及这些构件之间的一致性。变化的管理，保证了对系统的修改始终与项目目标相一致。
- 第 14 章“项目管理”描述了一些在开启一个软件开发项目、跟踪项目进度以及处理所遇到的风险和意外事件时所必需的技巧。我们主要关注那些有大批参与者合作并在计划期限内交付高质量系统所涉及的组织、角色和管理活动。
- 第 15 章“软件生命周期”描述了提供开发活动抽象模型的软件生命周期，例如 Boehm 的螺旋模型和统一的软件开发过程。在本章中，我们还讨论了能力成熟度模型，该模型用来评价相关开发组织的成熟性。
- 第 16 章“方法学：综合考虑各种因素”描述了将上述章节中所描述的内容应用于具体情况的方法学和启发式准则。无论需求获取做得怎样彻底，也无论计划多么详细，任何规模的实际项目都将遇到预料之外的事件和变化。不确定性处理使得实际项目和系统与我们在书本上讨论的项目和系统差别极大。在本章中，我们描述了很多不同的方法学，讨论了在每一个项目中都需要解决的问题，提供了三个实际项目的实例分析。

上面这些主题是密切相关的。为了强调它们之间的关系，我们选择了一种重复的方法。每一章都由 5 节组成。第 1 节用一个阐述性实例介绍与主题相关的问题。第 2 节简要描述

了与这些主题相关的活动。第 3 节通过一些简单的实例来解释主题的一些基本概念。第 4 节使用真实系统中的例子来详细讲解技术活动。最后，我们讲述管理活动并讨论一些典型的交易。在第 4～10 章，我们对一个称为竞技场的、复杂的多用户游戏管理系统进行了持续的实例分析，通过越来越复杂的实例来重复和详述同一类概念，我们希望能够提供给读者面向对象软件工程的操作知识。

课程

构造一个大型、复杂系统可以比作攀登一座大山，需要有路线描述。但是路线永远不可能完全被勾画出来，因为新的例外随时会出现。尽管我们在本书中绘制了软件工程知识图，变化还会发生，而且我们现在所坚信的方法也可能很快变得过时了。

我们怎样教会我们的学生去处理这类快速变化的情况？对我们而言，我们应该传授给学生的最重要的不仅是知识体系，而且还应该教会他们成功掌握领域知识的能力。尽管学习路线描述是明智的，但是没有什么能够取代在该路线上进行实际旅行的经验。

本书是为本科毕业班的学生和研究生编写的，适合作为一个学期的软件工程课程的教材。学生应该具有一定的使用 C、C++、C＃或 Java 进行编程能力。我们希望学生具有必要的解决问题的技能来解决技术问题，但无需接触过系统开发中常见的复杂多变情况。此外，本书也可以用于支持其他类型的课程，例如短期密集的专业课程培训等。

项目和高级课程

一门项目课程应包括本书中的所有章节，大概按顺序讲授。教师可以考虑在介绍课程时，提前介绍第 14 章“项目管理”中的更高深的项目管理概念，以让学生熟悉制定计划和控制状况。

介绍性课程

一门有作业的介绍性的课程应该把重点集中在每一章的前面 3 节上。第 4 节以及实例分析可以用做作业的材料，也可以通过使用纸质 UML 图表、文档和编码来模拟构造一个微型系统。

简短（短期）技术课程

本书也可以用来作为短期且密集的专业培训指导。一个技术性的课程，其重点在于 UML 和面向对象的方法方面，所涉及的部分章节按顺序为第 1、2、4、5、6、7、8、9、10、11 章，包括了从需求获取到测试的所有开发阶段。高级课程还应该包含第 12 章“基本原理管理”、第 13 章“配置管理”。

简短（短期）管理课程

本书还可以用作旨在面向管理者的短期密集培训课程。管理课程的重点集中在沟通、风险管理、基本原理、成熟度模型和 UML 等管理方面，可选用的章节按顺序有第 1、2、3、14、15、16、12、13 章。

自第 2 版所作的修改

这次新版在第 2 版的基础上进行了更新并结合了 UML 2 以及敏捷方法的最新进展。其间我们增加了系统设计和测试的新材料。非常感谢我们的出版人 Tracy Dunkelberger 的耐心。我们所作修改如下：

- 全面提升到最新的 UML 以及 OCL 标准。我们修改了本书中大部分的图表，以利用

UML 和 OCL 最新的进展。特别是我们系统和对象设计中使用了有球窝记号的构件图。

- 扩展了敏捷方法的资料。在第 2 版中，我们在第 16 章引入了 XP 技术。在这一版本中，我们把敏捷材料扩展到 Scrum 和 Rugby，并在第 11、13 和 14 章中用于测试、软件配置管理和项目管理上。
- 增加了连续集成的新内容。也用于其他场合的敏捷方法的一种做法是不断地把软件变化集成到产品主线上。尽管这一做法能够找到并因此解决集成问题，但其实现也提出了许多挑战。我们在第 13 章“配置管理”中提出了这一新内容。
- 增加了关于 U2TP 和自动测试的新内容。在我们的教学中我们发现 UML 2 测试概况的扩展有助于讨论测试概念，特别是有助于区分测试系统和待测系统。这也使得我们得以把测试材料扩展到自动测试和自动化测试生成。
- 改进了贯穿始终的案例研究和实例。自上一版以来，我们收到了大量关于本书的案例研究和实例的反馈意见。我们感谢这些反馈意见并最终实现了一些建议，这些建议太多了无法在这里逐一列举。

关于作者

Bernd Bruegge 博士在卡耐基·梅隆大学（CMU）研究并教授软件工程课程长达 20 年之久，在 CMU，B. Bruegge 获得了硕士和博士学位。Bruegge 博士还获得了德国汉堡大学的毕业证书。目前，Bruegge 博士是 Technische Universität München 计算机科学教授，负责应用软件工程，同时是 CMU 的客座教授。对本书中所描写的面向对象软件工程项目课程，Bruegge 博士进行了长达 15 年的教学。他于 1995 年在 CMU 获得 Herbert A. Simon 优秀教学奖。Bruegge 博士还是一位国际顾问，他使用了本书中介绍的技术，设计并实现了很多应用系统，有 DaimlerChrysler 工程反馈系统、美国环境保护处的环境建模系统、市警察局的事故管理系统等。

Allen Dutoit 博士从事航空工业领域的航空电子技术软件开发，是 Technische Universität München 的研究科学家。Allen Dutoit 博士在 CUM 获得了硕士和博士学位，在瑞士联邦技术学院获得工学毕业证书。从 1993 年开始，Allen Dutoit 博士与 Bruegge 教授一起，在 CUM 和 Technische Universität München 大学教授软件工程课程，在这期间他们使用并完善了本书中所描述方法。Allen Dutoit 博士的研究包含了软件工程和面向对象系统的多个方面，涉及需求工程、基本原理管理、分布开发系统和基于原型开发的系统等。Allen Dutoit 博士曾经是 CMU 软件工程学会和复杂工程系统学会的会员。

致　谢

本书的编写过程十分复杂，并经过了多次修改。在1989年，本书的第一作者（B.B.）开始以单项目课程的形式讲解软件工程。目的是通过解决真实约束下由真实客户使用真实工具描述的真实问题的方法，向学生揭示软件工程中的重要问题。在卡耐基·梅隆大学的第一次开课课程表中的编号是15-413，总共有19名学生学习了该课程，在学习过程中使用了系统分析/系统设计，并产生了4000行左右的代码。由于受到James Rumbaugh和他的同事在面向对象建模和设计方面所编写的书籍的深刻影响，从此我们开始使用面向对象方法。我们在卡耐基·梅隆大学和慕尼黑工业大学开设了一些分布式项目课程，总共有接近100名学生学习该课程，并得到了一个包含有500页文档和50000行代码的系统。现在我们教授来自新西兰Otago大学和慕尼黑工业大学的学生，以完成一个分布式项目课程。

项目课程的缺点是教师不能避免课程的复杂性，并提升学生的经验。为此，教师也应很快成为了开发团队的一员，而且通常是担当项目经理之类的角色。我们希望这本书能够帮助教师和学生克服这个过程的复杂性和变化。

尽管在该课程上花费了很多精力，我们还是找到了时间来编写和完成这本书及其后继修订版本，这都要归功于众多学生、客户、助教、支持人员、教师、评论者、Prentice Hall的职员以及我们家庭的帮助和耐心。这中间有些人帮助我们提高课程的授课水平，有些人为我们的原稿提供了建设性的反馈意见，另外一些人则给予我们极大的鼓励。在过去的20年中，我们欠了我在致谢中提到的很多人的人情，谢谢他们给予的帮助。

项目课程的参与者：Workstation Fax（1989），Interactive Maps（1991），Interactive Pittsburgh（1991），FRIEND（1992、1993、1994），JEWEL，GEMS（1991、1994、1995），DIAMOND（1995、1996），OWL（1996、1997），JAMES（1997、1998），PAID（1998、1999），STARS（1999、2000、2001），TRAMP（2001、2002），ARENA（2002、2003）。

项目的支持者：谢谢为项目做出贡献、在我们需要的时候使我们脱离麻烦善良的人们：Martin Bauer，Ulrich Bauer，Catherine Copetas，Oliver Creighton，Ava Cruse，Barry Eisel，Luca Girardo，Dieter Hege，Mariss Jansons，Joyce Johnstone，Rafael Kobylinski，Asa MacWilliams，Monika Markl，Pat Miller，Martin Ott，Ralf Pfleghar，Martin Pittenauer，Barbara Sandling，Christian Sandor，Ralph Schiessl，Arno Schmackpfeffer，Helma Schneider，Stephan Schoenig，Steffen Schwarz，Martin Wagner，Uta Weber，Timo Wolf。

同事、教师、影响过我们的朋友们：Mario Barbacci，Len Bass，Ben Bennington，Elizabeth Bigelow，Roberto Bisiani，Naoufel Boulila，Harry Q Bovik，Andreas Braun，Manfred Broy，Sharon Burks，Marvin Carr，Mike Collins，Robert Coyne，Douglas Cunningham，Michael Ehrenberger，Kim Faught，Peter Feiler，Allen Fisher，Laura Forsyth，Eric Gardner，Helen Granger，Thomas Gross，Volker Hartkopf，Bruce Horn，David Kauffer，Gudrun Klinker，Kalyka Konda，Suresh Konda，Rich Korf，Birgitte Krogh，Sean Levy，Frank Mang，

K.C.Marshall，Dick Martin（“Tang Soo”），Horst Mauersberg，Roy Maxion，Russ Milliken，Ira Monarch，Rhonda Moyer，Robert Patrick，Brigitte Pihulak，Mark Pollard，Martin Purvis，Raj Reddy，Yoram Reich，James Rumbaugh，Johann Schlichter，Mary Shaw，Jane Siegel，Daniel Siewiorek，Asim Smailagic，Mark Stehlik，Eswaran Subrahmanian，Stephanie Szakal，Tara Taylor，Michael Terk，Günter Teubner，Marc Thomas，Walter Tichy，Jim Tomayko，Blake Ward，Alex Waibel，Art Westerberg，Jeannette Wing，Tao Zhang。

给予我们建设性意见的评阅人和帮助我们更正许多细节的人：Martin Barrett，Thomas Eichhorn，Henry Etlinger，Ray Ford，Gerhard Mueller，Michael Nagel，Barbara Paech，Joan Peckham，Ingo Schneider，还有许多匿名的读者提供了建设性和细节上的评论。本书中所有依然存在的错误都是我们自己造成的。

Prentice Hall 的所有工作人员：他们使得这本书得以付印，特别要感谢本书的发行人：Alan Apt，Lakshmi Balasubramanian，Toni Holm，Patrick Lindner，Camille Trentacoste，Jake Warde，以及为该书的完成付出了艰辛劳动但我们却没有机会和运气认识的人们。

最后，要感谢我们的家庭，没有他们无限的爱和耐心，要完成此书是不可能的，因此我们将此书献给他们。

目　录

译者的话 ····· I
前言 ····· III
序言 ····· V
致谢 ····· XI

第 1 部分　开　始

第 1 章　软件工程导论 ····· 3
1.1　导言：软件工程的失误 ····· 3
1.2　什么是软件工程 ····· 5
1.2.1　建模 ····· 6
1.2.2　问题求解 ····· 7
1.2.3　知识获取 ····· 8
1.2.4　基本原理 ····· 8
1.3　软件工程概念 ····· 9
1.3.1　参与者与角色 ····· 9
1.3.2　系统与模型 ····· 10
1.3.3　工作产品 ····· 11
1.3.4　活动、任务与资源 ····· 11
1.3.5　功能性需求与非功能性需求 ····· 12
1.3.6　记号、方法和方法学 ····· 12
1.4　软件工程开发活动 ····· 13
1.4.1　需求获取 ····· 13
1.4.2　分析 ····· 14
1.4.3　系统设计 ····· 16
1.4.4　对象设计 ····· 16
1.4.5　实现 ····· 16
1.4.6　测试 ····· 17
1.5　管理软件开发 ····· 17
1.5.1　沟通 ····· 17
1.5.2　基本原理管理 ····· 18
1.5.3　软件配置管理 ····· 18
1.5.4　项目管理 ····· 18
1.5.5　软件生命周期 ····· 19

1.5.6 总结……19
1.6 竞技场实例分析……19
1.7 推荐读物……20
1.8 练习……21
第 2 章 使用 UML 进行建模……22
2.1 导言……22
2.2 UML 综述……23
2.2.1 用例图……23
2.2.2 类图……24
2.2.3 交互图……25
2.2.4 状态机……25
2.2.5 活动图……26
2.3 建模活动中的概念……27
2.3.1 系统、模型和视点……27
2.3.2 数据类型、抽象数据类型和实例……29
2.3.3 类、抽象类和对象……29
2.3.4 事件类、事件和消息……31
2.3.5 面向对象建模过程……31
2.3.6 约简表达和原型构造……33
2.4 UML 的深入透视……34
2.4.1 用例图……34
2.4.2 类图……39
2.4.3 交互图……47
2.4.4 状态机……49
2.4.5 活动图……51
2.4.6 图的组织……53
2.4.7 图的扩展……55
2.5 推荐读物……56
2.6 练习……56
第 3 章 项目组织和沟通……58
3.1 引言：一个有关火箭的例子……58
3.2 项目综述……59
3.3 项目管理概念……62
3.3.1 项目管理……62
3.3.2 角色……65
3.3.3 任务和工作产品……67
3.3.4 进度表……68
3.4 项目沟通中的概念……69
3.4.1 计划内沟通……69

3.4.2 计划外的沟通……74
3.4.3 沟通机制……77
3.5 有组织的活动……83
3.5.1 加入一个项目团队……83
3.5.2 使用沟通基础设施……84
3.5.3 参加项目团队情况通气会议……84
3.5.4 组织客户和项目总结……86
3.6 推荐读物……87
3.7 练习……87

第 2 部分 复杂性处理

第 4 章 需求获取……91
4.1 引言：可用性实例……91
4.2 对需求获取的总的看法……92
4.3 需求获取概念……94
4.3.1 功能需求……94
4.3.2 非功能性需求……94
4.3.3 完全性、一致性、清晰性和正确性……95
4.3.4 现实性、确认和可追踪性……96
4.3.5 绿地工程、再工程和界面工程……97
4.4 需求获取活动……97
4.4.1 标识参与者（Actor）……98
4.4.2 标识场景……99
4.4.3 标识用例……101
4.4.4 求精用例……103
4.4.5 标识参与者和用例之间的关系……105
4.4.6 标识初始的分析对象……108
4.4.7 标识非功能性需求……110
4.5 需求获取管理……111
4.5.1 与客户协商规格说明：联合应用设计……112
4.5.2 追踪性维护……113
4.5.3 需求获取的编档……114
4.6 ARENA 实例研究……115
4.6.1 初始问题陈述……115
4.6.2 标识参与者和场景……117
4.6.3 标识用例……120
4.6.4 求精用例与标识关系……122
4.6.5 标识非功能性需求……126