面向对象技术实践丛书

HZ BOOKS

Prentice Hall

# C++

（原书第2版）

# 面向对象开发

*UML and C++:*
*A practical Guide to*
*Object-Oriented Development*
*(Second Edition)*

（美）Richard C. Lee William M. Tepfenhart 著
麻志毅 蒋严冰 译

机械工业出版社
China Machine Press

面向对象技术实践丛书

# C++面向对象开发

（原书第2版）

（美） Richard C. Lee
William M. Tepfenhart
贝尔实验室 著

麻志毅 蒋严冰 译

机械工业出版社
China Machine Press

面向对象技术是20世纪90年代对软件的最大发展，它不但改变了建构软件的方式，也改变了设计的方式，而C++则实现了面向对象的主要机制。因此，本书将面向对象技术与C++相结合，并使用UML这一可视化描述的建模语言进行表述。本书的目标是：通过应用面向对象技术和方法的基本原理来指导读者在使用C++开发软件和编程时适当考虑使用合适的面向对象的概念和良好的设计原则。

本书还附有两个完整的实践案例，在案例中，使用特定的面向对象技术，并应用面向对象的基本概念就一个项目的面向对象分析、设计和编程的各个阶段展开讨论，详细阐述了如何使用C++根据设计的模型进行编程，同时使读者深入地掌握和理解面向对象技术的应用。



**本书版权登记号：图字：01-2002-2189**

**图书在版编目（CIP）数据**

C++面向对象开发（原书第2版）/（美）李（Lee, R. C.），（美）特普芬哈特（Tepfenhart, W. M.）著；麻志毅等译. -北京：机械工业出版社，2002.9

（面向对象技术实践丛书）

书名原文：UML and C++：A Practical Guide to Object-Oriented Development，Second Edition

ISBN 7-111-10578-8

Ⅰ. C… Ⅱ. ① 李… ② 特… ③ 麻… Ⅲ. ① 面向对象技术，UML ② C++语言－程序设计 Ⅳ. TP312

中国版本图书馆CIP数据核字（2002）第051856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：杨海玲

北京第二外国语学院印刷厂印刷·新华书店北京发行所发行

2002年9月第1版第1次印刷

787mm × 1092mm 1/16 · 28.75印张

印数：0 001- 4 000册

定价：45.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

# 译者序

目前，计算机软件界的各方人士都已经认识到了面向对象技术的优越性，大多数较先进的软件开发组织在分析、设计、编程和测试阶段都全面地采用了面向对象技术。面向对象技术无疑已经成为当前软件领域的主流技术。

UML是一种用于对软件密集型系统进行可视化描述、构造和文档化的建模语言，主要适用于分析与设计阶段的系统建模。UML的问世受到计算机软件界的广泛重视，因为它代表了一个积极的方向，即多种方法相互借鉴、相互融合、趋于一致和走向标准化。UML最主要的特点是表达能力很强，可以说，UML对系统模型的表达能力超出了以往任何一种面向对象分析和面向对象设计方法。学习和使用UML已经成为一种潮流，许多研究人员和技术人员已在数年前开始学习和研究UML。

流行的面向对象程序设计语言有很多，C++是其中的佼佼者。它不但实现了面向对象的主要机制，在一些有意义的方面也改善了C，并与C是高度兼容的。此外，C++是一种多范型语言，支持多种编程范型。

目前，面向对象编程方面的书籍很多，面向对象理论方面的书籍也正在陆续面市，但有关面向对象的理论与实践相结合方面的书籍还不多见，而这样的书籍又恰恰是广大从事软件开发的人士所迫切需要的。本书在面向对象的理论和实践上结合得很好，为了使中国的读者能够更好地从中受益，我们受机械工业出版社委托，翻译了本书。

在理论方面，本书阐明了面向对象技术的基本概念和面向对象的基本原理；在实践方面，提供了面向对象分析和面向对象设计的实用方法，其中对用面向对象技术进行软件开发的步骤还给出了指导，并较为详细地阐述了如何使用C++根据所设计的模型进行编程。此外，本书给出的两个较为完整的开发实例，能帮助读者深入地掌握和理解怎样应用面向对象技术。这在其他的同类书籍中也是少见的。

由于译者自身的知识局限，而且时间也比较仓促，译文难免存在着错误和疏漏，诚恳地希望读者给予批评与指正。

译 者

2002年1月于北京

## 译者简介

麻志毅，男，北京大学计算机科学技术系副教授。现已发表学术论文40余篇，主持或参加政府科研项目十余项。主要研究领域为软件工程、面向对象技术和计算语言学。

蒋严冰，男，2000年考入北京大学计算机科学技术系攻读博士学位。主要研究方向为面向对象技术、软件工程环境和软件构件技术。

# 第2版序言

哲学中的公理并不是公理，除非它们与我们息息相通：我们读到好的文章却从没完全领会它们，除非我们同作者心心相印。

约翰·济慈（1795—1821），
致乔舒亚·雷诺兹的信，1818年5月3日

《C++面向对象开发》仍然是为那些为建立大型系统繁忙工作的专业软件分析人员和开发人员编著的，特别是那些需要将他们的新系统与旧系统结合起来的开发人员写的。如果你仍然没有抽出时间参加培训班来跟上使用统一建模语言(UML)和C++这些面向对象（OO）技术的发展的话，本书就是你的自学指导。它将帮助你理解面向对象的分析、设计和编程之间的区别。在第1版中我们的目标是：

- 教读者如何使用C++构造面向对象的应用，以及为满足业务需要如何做出正确的选择。
- 阐明与面向对象技术相关的基本概念。
- 为进入该领域并跟上其发展步伐的学生和实践者提供足够深入的知识。
- 揭示围绕OO技术的一些神秘性，并集中阐述它作为软件工程工具的实用性。
- 对于怎样执行面向对象技术的各个步骤，给出一个“处方”或者说一步一步的指导。
- 提倡这样的观点：将OO、基于规则的概念、模糊逻辑、多媒体和数据建模融合成单一的模型，足以应付目前和长远的对信息技术组织的行业挑战。
- 为面向对象技术中的分析、设计和编程提供实用方法。
- 说明怎样使用C++实现面向对象技术（C++尽管不是面向对象的语言，却是一种特别强大的多范型语言）。
- 衡量现有的文献中的理论和应用实践。

在第2版中，这些目标扩大到：

- 对作为面向对象分析的一部分的用况开发，给出一套实用方法。
- 在更大的范围内提供了UML图示方式。
- 介绍C++中主要的C++类库，它们为支持C++实现面向对象模型提供了重要功能。

此外，第2版对下列的题材做了改进：

- 动态行为建模。
- 状态模型的实现。
- 类工程。

同以往一样，不需要计算机科学或高深的数学知识你也能深入理解重要的面向对象概念和问题。即使在与编程有关的各章中也不要求有C++的背景；这些章说明C++的工作代码是怎样产生的。

## 面向对象技术

作为大型系统的软件开发者，我们认为面向对象技术是20世纪90年代对软件的最重要发展（变革）。它改变了我们构建软件的方式，也改变了全球范围内的网络和多厂商的计算机上的应用程序相互通信的方式。此外，面向对象的模型也改变了我们设计业务过程的方式和对企业的思考方式。大多数企业现在需要重新设计，以应付未来的行业挑战。

业务过程的再设计是信息技术组织最重要的功能之一。捕获业务过程、步骤、政策和规则的模型可以简化设计。将模型转化为可操作的系统的工具加速了再设计的实现。当市场或业务条件发生改变时，应该通过更新模型并使用这些工具重新产生这些系统，以反映变化。在以前的几十年中，与其他方法相比，信息科学（工程）使我们走得更远和更快。然而，它需要不断革新，通过更好、更精致的方法来应付由面向对象建模和编程带来的需求和挑战。越来越多的人认为面向对象技术将缓解软件危机，这意味着面向对象的机制对软件来说，就像立柱和横梁对建筑设计以及芯片对计算机硬件设计一样重要。这一信条基于以下因素：

- 精通更高层次的面向对象模型可以为软件设计者提供现实的和可编程的构件，因此减少了软件开发费用。
- 使用面向对象技术来共享和复用代码将减少开发应用程序的时间。
- 通过程序抽象机制，面向对象技术具有将修改的影响局部化和最小化的能力，这种能力允许进行更快的增值开发，并且可提供更可靠和更健壮的软件。
- 面向对象的管理复杂性的能力允许开发者开发更困难的应用。

面向对象的概念集是对现实世界建模的工具集。这个工具集为开发者提供最好的管理复杂性的方法。一些OO概念可以帮助开发者生产具有灵活性和可维护性的软件。

## 为什么使用统一建模语言

作为面向对象的实践者，我们认为所有的方法如果应用适当，都会产生相同或相似的模型。多年以来，建模语言的表示法的不同阻碍了软件的发展，我们在UML被广泛地接受或被建模工具支持之前就采纳了它。对UML广泛地采纳，在很大程度上排除了这样的问题——建模语言表示法的差异以及缺少建模工具。

统一建模语言成功的基本原因是：在解决实际业务问题时，它为我们提供了所有的对捕获我们认为有价值的概念或机制有必要的作图图标。同时，它也提供了对模型文档化至关重要的图。此外，它是一种有活力的语言，为我们提供了作为机制的扩展表示法的能力，而这些表示法尚未被Rational软件公司著名的Grady Booch、James Rumbaugh和Ivor Jacobson小组所定义。

## 为什么使用C++

认为C++是单纯的面向对象语言的观念是错误的。它是一种多范型语言，支持很多编程范型，

包括过程的、抽象数据类型的和面向对象的范型。我们将向你展示怎样使用C++中的面向对象范型将你的面向对象的模型映射成为C++的结构。我们也将向你展示在面向对象设计的上下文中怎样使用C++的非面向对象的概念，以帮助你满足你的业务需要。

我们选择C++作为我们的语言有两个实际原因。第一，也是最重要的，大多数开发者必须解决：与旧工程的接口以及在数据库、存储和性能方面的技术限制。C++为开发者提供了多种范型，使他们可以根据需要进行调整。第二，开发商已经把钱投到了C++上，提供了一些工具和编译器。

## 我们的面向对象技术的方法

我们并不是纯粹的面向对象追随者，也不是理论家。我们是愿意使用任何好的思想的开发人员，只要它们能帮助我们达到两个十分重要的业务目标：更低的开发费用和更短的提交到市场的时间。我们认为可靠性、可维护性和灵活性这几项技术指标对达到上述两个业务目标是重要的。

我们使用面向对象技术方法是要管理软件开发中的复杂性，使软件可靠、可维护并具有灵活性。管理复杂性是达到这些目标（也就是业务目标）的关键。为管理复杂问题域中的复杂性，我们发现开发者需要知道对象、类、关系和规则是怎样融入到对象范型中的。对大多数复杂的问题域建模时，我们要发现对象、类以及对象间的很多关系。此外，我们需要捕获在问题域中的规则（原则）。因此，我们必须使用十分丰富的静态建模技术来捕获数据（对象）关系。

许多面向对象的专家认为关系是“坏的”，因为它们破坏了封装原则。从我们的观点看，它帮助我们管理问题域的复杂性，并帮助我们达到业务目标。我们愿意使用它，并且要寻找支持这一方面的更多的机制和语言。在写第 9 章的说明语义时，我们认为应把规则和原则作为模型中一个整体部分，而不应写成特殊子系统的扩展。

使用机制帮助我们对复杂的问题域建模，同选择UML作为我们的建模语言和C++作为我们的编程语言是一致的。UML和C++可以定义任何帮助我们构造更加可管理的软件的机制。

我们讨论行为（动态的和静态的）和多态来捕获模型的过程性方面的特征。在处理定时、同步和中断时，使用有限状态机或其他的状态模型来帮助我们管理过程的复杂性。这些领域在大多数面向对象的书中被忽略了或被忽视了。我们并没有深入地探究这些问题，只是为使读者能使用加入到UML中的语义而打下基础。

我们认为构造大型面向对象系统的成功的关键是要求开发者和程序员的知识比大多数面向对象的书中所教的要多。构造大型的系统，要求应用由一些面向对象的专家所倡导但却没有被一致接受的机制。专业开发人员在成为多产的团队成员之前，至少需要理解问题域的这些方面应该怎样被处理。本书不会使你成为一名专家，你仍然需要专家/顾问来开发系统。通过应用80/20规则，本书提供可以使你高产的80%和理解专家是怎样解决困难的20%。

在本书中，我们没有涵盖面向对象技术最新的趋势和潮流，包括对象设计模式和分布式对象计算。虽然这些内容是有趣的，但是在为开发新手提供实践框架以便尽快使其进入从事OO编

程，我们并不认为它们能有什么明显的帮助。设计模式被证明是可以增进良好的面向对象实践的技术，但其本身并不是要成为一种范型。我们仍然认为，学好面向对象的基础是正确应用设计模式的根本。通过编写良好的中间件，使分布式计算尽可能对应用程序的开发者透明。我们期待着基础系统开发商最终将把这项技术透明地提供给应用领域。

最后，我们并不同意大多数专家所说的面向对象技术是一种成熟的技术。我们认为它还处于其幼年期；使我们产生强烈印象的是用这项不成熟的技术能够取得多大的成就。面向对象技术具有早期的技术（过程的、功能的和基于规则的等）所不具备的巨大的潜能，它能帮助我们管理复杂性。

## 本书的组织

我们通过应用面向对象技术和方法的基本原理来指导读者。这并不是一套抽象的规定。我们的目标是教你在使用C++开发软件和编程时考虑使用合适的面向对象的概念和良好的设计原则。在案例分析中，我们使用特定的面向对象技术并应用面向对象的基本概念，就一个项目的面向对象的分析、设计和编程的各个阶段展开讨论。用C++实现设计，发挥C++的技术性能。

本书最初是以可以顺序阅读的方式编写的，以便于自学。我们保留了本书的这一特征。每一章都讨论了我们的面向对象技术方法的一个主要步骤。大多数章节都包含了一步一步的指导或方法。我们希望读者将这些步骤仅作为指导使用；依靠常识而不要盲目地遵循所提出的步骤。

**第1章** 提供为什么众多公司对面向对象感兴趣以及为什么软件专业人员应该了解面向对象的理由。

**第2章** 描述软件业务以及管理复杂性的需要。

**第3章** 描述怎样找出面向对象技术中的基本术语和关键概念。

**第4章** 描述怎样使用用况将领域与相关的对象绑定。

**第5章** 描述怎样找出潜在的对象，这是应用面向对象技术的第一步。

**第6章** 通过识别与对象相关的属性（数据）和服务，描述怎样区分“真”的对象和“假”的对象。

**第7章** 说明怎样捕获对象的行为。

**第8章** 讲述怎样识别和描述动态行为。

**第9章** 讲述为组织系统中的所有对象所用到的不同的关系（泛化/特化、链和对象聚合等）。

**第10章** 描述怎样将说明的事实结合到有关对象知识和一种基于规则机制的面向对象的模型中，以求实现这些事实。

**第11章** 讲述为利用C++中的机制怎样将对象变为类。

**第12章** 讲述面向对象系统开发中的一些设计问题[⊖]。

**第13章** 为使用C++进行面向对象编程提供了C++的基本知识。（熟悉C++的读者可跳过本章。）

---

⊖ 设计是十分复杂的，它是一个独立于面向对象技术的课题。

**第14章** 教你怎样实现一个类，类是创建对象的模板。
**第15章** 介绍在以后的几章中用到的C++类库。
**第16章** 教给你怎样实现第6章建立的行为规约。
**第17章** 教给你怎样实现第8章建立的动态行为。
**第18章** 讲述怎样使用C++中的类机制创建和销毁对象。
**第19章** 讲述怎样使用C++中的类派生机制实现泛化/特化（面向对象的关键概念之一）。
**第20章** 讲述怎样实现C++中不支持的其他关系。
**第21章** 介绍两个案例分析。
**第22章** 给出一个基于突围（Breakout）游戏的案例分析。
**第23章** 给出一个关于微波炉的案例分析。
**附录A** 概述统一建模语言。

## 本书作为教材的用法

本书曾作为两门培训课程的教材，参加培训的是不懂面向对象和C++但有能力的程序员，在以一周为周期的两门课程中教授。第一门课程覆盖本书的前13章，第二门课覆盖本书其余各章。在第二门课程结束时，学生们已经有一个具备完整功能的程序，这个程序反映了第一门课程最后的设计。本书后面的案例分析就是学生们在这门课程中实现的项目。本书继续支持这种教学方式。

目前，本书的作者之一向已经懂C++的大学生讲授一门一学期的课程，使用本书前13章。他将在一个相关的项目中继续使用这部分材料，依靠项目本身而不是依靠作业来加深对概念的理解。使用有意义的项目使学生可以应用他们所学的概念。亲身经历一个项目的面向对象分析和设计的全过程看起来比做大量不连贯的和不相关的作业要好得多。有意义的分析和设计模型可以在第一个学期完成，第二学期有必要实现这个模型。

选择适当规模的项目可以使学生掌握所有的关键概念。教师Bill Tepfenhart布置了一个大型的探险游戏作为项目，允许学生们根据自己的主旨进行开发。在完成的项目中通常都包含100多个类和许多关系。这些游戏一般都包含了许多不同种类的地形、武器、妖怪、财宝和人物。某些游戏的复杂性已相当于许多商用游戏产品。

一个合理的项目组由三四个学生组成。更大的项目组在取得游戏主题的共识上，会花太多的时间，而更小的项目组就显得力不从心。这种规模的小组在更深层次上的优点是，可以使每个小组成员获得一种真正作为开发小组的一分子而进行工作的感觉。为了在所计划的时间内完成该项目，他们必须每周都工作。

下面是建议的授课计划。它假定需要15周，并在第15周进行最终测验。本计划的关键特征是它为定义游戏和开发用况留出了充裕的时间。上课指导每个小组的活动需要3周的时间。这已经被证实是有益的，因为它可以帮助学生们避免犯一些常见的错误，例如混淆对象状态和对象属性。

表1 面向对象分析和设计的课时安排

| 周 | 章 | 项目活动 |
|---|---|---|
| 1 | 1，2 | 分组 |
| 2 | 3 | 开发游戏思路 |
| 3 | 4 | 开始写用况 |
| 4 | 5 | 继续写用况 |
| 5 | 6 | 继续写用况 |
| 6 | 7 | 识别对象和对象的属性 |
| 7 | 8 | 识别对象的静态行为 |
| 8 | 9 | 识别动态行为 |
| 9 | 10 | 识别关系 |
| 10 | 11 | 评审模型 |
| 11 | 12 | 设计 |
| 12 | 13 | 设计 |
| 13 | 评审 | 完成项目 |
| 14 | — | 课堂介绍 |

## 致谢

我们感谢许多人。本书的动力来自上Richard Lee的课程的数以千计的学生，以及他的朋友们和他的许多同事。他感谢他们不顾一切的支持和鼓励。

因为我们是开发者（而不是研究者、学者或作者），我们以面向对象研究者（所有这些思想来源于他们）和面向对象作者（他们在较早的著作中将思想传给了我们）的工作作为支撑。我们只是以实用的方式，使用这些思想构造实际的应用。我们对这些思想、概念、机制和技术的提出者和在我们之前的许许多多面向对象的作者致谢，感谢他们，没有他们就不可能有这本书。

理论和思想是美妙的；然而对实践者来说，经验是最好的老师。如果我们没有在实际项目中应用面向对象技术和方法的经验，就不可能产生这本书。感谢我们的许多有勇气让我们做前沿（许多次是痛苦的边沿）的软件开发的老板，无论是现在的还是过去的。没有他们的支持，就不能测试我们所写的东西。

Richard Lee感谢众多为他工作并在将本书中所写的思想应用到实际项目过程中充当先锋的人们。他们和他分享着“第一次”将面向对象技术应用到各自公司的大项目中的兴奋和悲伤。（或者说，他们遵循他的领导的要求并接受公司中没有其他人想面对的挑战是不是愚蠢的？）Richard对他们致以深深的谢意。

William Tepfenhart感谢他的合作伙伴，无论是过去的还是现在的，他们中的每一个人都对他理解计算机科学有过帮助。他感谢 Bill Case、Freid Elliot 和 Dayton Eden，他们使他从写物理系统的FORTRAN模型转变到关于物理系统推理的人工智能模型。他感谢将他的建模技术扩展到包括对象、关系和规则的同事们。

没有下面这些人的帮助，本书的第2版就不可能出版：

■ Terrance Cleary，AT&T实验室
■ Dan Dvorak，喷气推进实验室
■ John Eddy，AT&T实验室
■ Bruce Handelman，AT&T
■ Rolf Kamp，AT&T
■ David Lundin，pcorder.com
■ David Simen，AT&T

上述人员花了大量的时间和精力审阅了手稿。真诚地感激他们对本书的评价和见解。

本书得以出版，很大程度上还要归功于我们在AT&T和Bell实验室的经理Dick Machol、Raj Warty、Moses Ling和Raj Dube，深深地感谢他们的支持。如果他们不允许我们下班后使用计算机工具与资源，我们就不能完成本书。同时，也感谢蒙茅斯大学的管理人员，他们赞助了第2版的费用。然而，这不应解释为本书是被AT&T或Lucent Bell实验室认可的书。本书是我们对面向对象技术的个人看法，它主要基于Richard Lee在软件开发业务的35年的经验以及他对面向对象技术的实践。William Tepfenhart提供了很多优秀的“处方”，并且组织了材料，使得开发者可以快速地成为开发小组中有贡献的成员。

最后，我们对提供宝贵反馈意见的审阅者表示感谢和感激，他们是Rex Jaeschke（ANCI C委员会的主席，独立的顾问和作者）和Robert Taylor（Taylor计算公司）。作为作者，我们十分高兴接受所有有关错误、疏忽、不准确、不真实、思想模糊以及有关质量的批评。我们欢迎所有建设性的评论，并将愉快地忽略破坏性的评论。

Richard C.Lee
William M.Tepfenhart

# 第1版序言

哲学中的公理并不是公理，除非它们与我们息息相通：我们读到好的文章却从没完全领会它们，除非我们同作者心心相印。

约翰·济慈（1795—1821），
致乔舒亚·雷诺兹的信，1818年5月3日

《C++面向对象开发》是为那些为建立大型系统繁忙工作的专业软件分析人员和开发人员而编著的，特别是为那些需要将他们的新系统与旧系统结合起来的开发人员写的。如果你仍然没有抽出时间参加一个培训班来跟上使用统一建模语言(UML)和C++这些面向对象（OO）技术的发展的话，本书就是你的自学指导。它将帮助你理解面向对象的分析、设计和编程之间的区别。我们的目标是：

- 教读者如何使用C++构造面向对象的应用，以及为满足业务需要如何做出正确的选择。
- 阐明与面向对象技术相关的基本概念。
- 为进入该领域并跟上其发展步伐的学生和实践者提供足够深入的知识。
- 揭示围绕OO技术的一些神秘性，并集中阐述它作为软件工程工具的实用性。
- 对于怎样使用所有的面向对象技术，给出了一个“处方”或者说一步一步的指导。
- 提倡这样的观点：将OO、基于规则的概念、模糊逻辑、多媒体和数据建模融合到模型中，足以应付目前和长远的对信息技术组织的行业挑战。
- 提供面向对象技术的分析、设计和编程的实用方法。
- 说明怎样使用C++实现面向对象技术（C++尽管不是面向对象的语言，却是一种特别强大的多范型语言）。
- 衡量现有的文献中的理论和应用实践。

不用计算机科学或高深的数学知识你也能深入理解重要的面向对象的概念和问题。即使在与编程有关的各章中也不要求有C++的背景；这些章说明C++的工作代码是怎样产生的。

## 面向对象技术

作为大型系统的软件开发者，我们认为面向对象技术是20世纪90年代对软件的最重要发展（变革）。它改变了我们构建软件的方式，也改变了全球范围内的网络和多厂商计算机上的应用程序相互通信的方式。此外，面向对象的模型也改变了我们设计业务过程的方式和对企业的思

考方式。大多数企业现在需要重新设计，以应付未来的行业挑战。

业务过程的再设计是信息技术组织最重要的功能之一。捕获业务过程、步骤、政策和规则的模型可以简化设计。将模型转化为可操作的系统的工具加速了再设计的实现。当市场或业务条件发生改变时，应该通过更新模型并使用这些工具重新产生这些系统，以反映变化。在以前的几十年中，与其他方法相比，信息科学（工程）使我们走得更远和更快。然而，它需要通过更好、更精致的方法不断地更新来应付由面向对象建模和编程带来的需求和挑战。越来越多的人认为面向对象技术将缓解软件危机，这意味着面向对象的机制对软件来说，就像立柱和横梁对建筑设计以及芯片对计算机硬件设计一样重要。这一信条基于以下因素：

- 精通更高层次的面向对象模型可以为软件设计者提供现实的和可编程的构件，因此减少了软件开发费用。
- 使用面向对象技术来共享和复用代码将减少开发应用程序的时间。
- 通过程序抽象机制，面向对象技术具有将修改的影响局部化和最小化的能力，这种能力允许进行更快的增值开发，并且可提供更可靠和更健壮的软件。
- 面向对象的管理复杂性的能力允许开发者开发更困难的应用程序。

面向对象的概念集是对现实世界建模的工具集。这个工具集为开发者提供最好的管理复杂性的方法。一些OO概念可以帮助开发者生产具有灵活性和可维护性的软件。

## 为什么使用统一建模语言

作为面向对象的实践者，我们认为所有的方法如果应用适当，都会导致相同或相似的模型。多年以来，建模语言的表示法的不同阻碍了软件的发展。我们对能帮助我们以更低的花费、更短的时间产生更可维护的软件的结果感兴趣。在解决实际企业问题时，统一建模语言为我们提供了对捕获我们认为有价值的概念或机制所必需的所有的作图图标。同时，它也提供对模型文档化所必需的图。此外，它是一种有活力的语言，为我们提供了作为机制的扩展表示法的能力，而这些表示法尚未被Rational软件公司著名的Grady Booch、James Rumbaugh和Ivor Jacobson小组所定义。

## 为什么使用C++

认为C++是单纯的面向对象语言的观念是错误的。它是一种多范型语言，支持很多编程范型，包括过程的、抽象数据类型的和面向对象的范型。我们将向你展示怎样使用C++中的面向对象范型，将你的面向对象的模型映射成为C++的结构。我们也将向你展示在面向对象设计的上下文中怎样使用C++的非面向对象的概念，以帮助你满足你的业务需要。

我们选择C++作为我们的语言有两个实际原因。第一，也是最重要的，大多数开发者必须解决：与旧工程的接口以及在数据库、存储和性能方面的技术限制。C++为开发者提供了多种范型，使他们可以根据需要进行调整。第二，开发商已经把钱投到了C++上，提供了一些工具和编译器。

## 我们的面向对象技术的方法

我们并不是纯粹的面向对象追随者，也不是理论家。我们是愿意使用任何好的思想的开发人员，只要它们能帮助我们达到两个十分重要的业务目标：更低的开发费用和更短的提交到市场的时间。我们认为可靠性、可维护性和灵活性这几项技术指标对达到上述两个业务目标是重要的。

我们使用面向对象技术方法是要管理软件开发中的复杂性，使软件可靠、可维护并具有灵活性。管理复杂性是达到这些目标（也就是业务目标）的关键。为管理复杂问题域中的复杂性，我们发现开发者需要知道对象、类、关系和规则是怎样融入到对象范型中的。对大多数复杂的问题域建模时，我们需要发现对象、类以及对象间的很多关系。此外，我们需要捕获在问题域中的规则（原则）。因此，我们必须使用十分丰富的静态建模技术来捕获数据（对象）关系。

许多面向对象的专家认为关系是“坏的”，因为它们破坏了封装原则。从我们的观点看，它帮助我们管理问题域的复杂性并帮助我们达到业务目标。我们愿意使用它，并且要寻找支持这一方面的更多的机制和语言。在写第9章中的说明语义时，我们认为应把规则和原则作为模型中的一个整体，而不应写成特殊的子系统的扩展。

使用机制帮助我们对复杂的问题域建模，同选择UML作为我们的建模语言和C++作为我们的编程语言是一致的。UML和C++可以定义任何帮助我们构造更加可管理的软件的机制。

我们讨论行为（动态的和静态的）和多态来捕获模型的过程性方面的特征。在处理定时、同步和中断时，使用有限状态机或其他的状态模型来帮助我们管理过程的复杂性。这些领域在大多数面向对象的书中被忽略了或被忽视了。我们并没有深入地探究这些问题，只是为使读者能使用加入到UML中的语义而打下基础。

我们认为构造大型面向对象系统的成功的关键是要求开发者和程序员的知识比大多数面向对象的书中所教的要多。构造大型的系统，要求应用由一些面向对象的专家所倡导但却没有被一致接受的机制。专业开发人员在成为多产的团队成员之前，至少需要理解问题域的这些方面应该怎样被处理。本书不会使你成为一名专家，你仍然需要专家/顾问来开发系统。通过应用80/20规则，本书提供可以使你高产的80%和理解专家是怎样解决困难的20%。

在本书中，我们没有涵盖面向对象技术中最新的趋势和潮流，包括面向对象设计模式标准模板库和分布式对象计算。虽然这些内容是有趣的，但是在为开发新手提供实践框架以便尽快使其进入OO编程，我们并不认为它们能有什么明显的帮助。

最后，我们并不同意大多数专家所说的面向对象技术是一种成熟的技术。我们认为它还处于其幼年期；使我们产生强烈印象的是用这项不成熟的技术我们能够取得多大的成就。面向对象技术具有早期的技术（过程的、功能的和基于规则的等）所不具备的巨大的潜能，它能帮助我们管理复杂性。

## 本书的组织

我们通过应用面向对象技术和方法的基本原理来指导读者。这并不是一套抽象的规定。

我们的目标是教你在使用C++开发软件和编程时考虑使用合适的面向对象的概念和良好的设计原则。在案例分析中，我们使用特定的面向对象技术并应用面向对象的基本概念，就一个项目的面向对象的分析、设计和编程的各个阶段展开讨论。用C++实现设计，发挥C++的技术性能。

本书是以可以顺序阅读的方式编写和设计的，以便于自学。我们采用了Richard多年来教授面向对象的概念和基本技术时使用的方法；我们并不倡导将这一方法作为构建面向对象系统的方法。每一章都讨论我们的面向对象技术方法的一个主要步骤。大多数章都包含了一步一步的指导或方法。我们希望读者将这些步骤仅作为指导使用；依靠常识而不要盲目地遵循所提出的步骤。

**第1章** 提供为什么众多公司对面向对象感兴趣以及为什么软件专业人员应该了解面向对象的理由。

**第2章** 描述软件业务以及管理复杂性的需要。

**第3章** 描述怎样找出面向对象技术中的基本术语和关键概念。

**第4章** 描述怎样找出潜在的对象，这是应用面向对象技术的第一步。

**第5章** 通过识别与对象相关的属性（数据）和服务，描述怎样区分“真”的对象和“假”的对象。

**第6章** 说明怎样捕获对象的行为。

**第7章** 描述怎样识别和描述动态行为。

**第8章** 描述为组织系统中的所有对象所用到的不同的关系（泛化/特化、链和对象聚合等）。

**第9章** 描述怎样将说明的事实结合到一个与对象知识和基于规则的机制相关的面向对象模型中，以求实现这些事实。

**第10章** 描述我们为了利用C++中的机制怎样将对象变为类。

**第11章** 讲述面向对象系统开发中的一些设计问题[⊖]。

**第12章** 为使用C++进行面向对象编程提供C++的基本知识。（熟悉C++的读者可跳过本章。）

**第13章** 教你怎样实现一个类，类是创建对象的模板。

**第14章** 教给你怎样实现第6章建立的行为规约。

**第15章** 讲述怎样使用C++中的类机制创建和销毁对象。

**第16章** 讲述怎样使用C++中的类派生机制实现泛化/特化（面向对象的关键概念之一）。

**第17章** 讲述怎样实现C++中不支持的其他关系。

**第18章** 介绍一个案例分析示例，说明使用第4章到第17章中的方法。

**第19章** 展现第一个小组在对该案例进行对象分析时所使用的方法。

**第20章** 展现第二个小组在对该案例进行对象分析时所使用的方法。

**第21章** 说明在第20章中得到的对象模型是怎样转化为对象设计并用C++实现的。

---

⊖ 设计是十分复杂的，它是独立于面向对象技术的课题。

# 致谢

我们感谢许多人。本书的动力来自上Richard Lee的课程的数以千计的学生，以及他的朋友们和他的许多同事。他感谢他们不顾一切的支持和鼓励。

因为我们是开发者（而不是研究者、学者或作者），我们以面向对象研究者（所有这些思想来源于他们）和面向对象作者（他们在较早的著作中将思想传给了我们）的工作作为支撑。我们只是以实用的方式，应用这些思想构造实际的应用。我们对这些思想、概念、机制和技术的提出者和在我们之前的许许多多面向对象的作者致谢，感谢他们，没有他们就不可能有这本书。

理论和思想是美妙的；然而对实践者来说，经验是最好的老师。如果我们没有在实际项目中应用面向对象技术和方法的经验，就不可能产生这本书。感谢我们的许多有勇气让我们做前沿（许多次是痛苦的边沿）的软件开发的老板，无论是现在的还是过去的。没有他们的支持，就不能测试我们所写的东西。

Richard Lee感谢众多为他工作并在将本书中所写的思想应用到实际项目过程中充当先锋的人们。他们和他分享着“第一次”将面向对象技术应用到各自公司的大项目中的兴奋和悲伤。（或者说，他们遵循他的领导的要求，并接受公司中没有其他人想面对的挑战是不是愚蠢的？）Richard对他们致以深深的谢意。

William Tepfenhart感谢他的合作伙伴，无论是过去的还是现在的，他们中的每一个人都对他理解计算机科学有过帮助。他感谢Bill Case、Freid Elliot和Dayton Eden，他们使他从写物理系统的FORTRAN模型转变到关于物理系统推理的人工智能模型。他感谢将他的建模技术扩展到包括对象、关系和规则的同事们。

我们对Barry Peiffer、David Siemen、John Eddy 和Dan Dvorak所花费的时间并给予反馈表示感谢。作为技术编辑的Steve Ruder作了出色的工作。

本书得以出版，很大程度上还要归功于我们在AT&T和Bell实验室的经理Dick Machol、Raj Warty、Moses Ling和Raj Dube，深深地感谢他们的支持。如果他们不允许我们下班后使用计算机工具与资源，我们就不能完成本书。然而，这不应解释为本书是被AT&T或Lucent Bell实验室认可的书。本书是我们对面向对象技术的个人看法，它主要基于Richard Lee在软件开发业务的35年的经验以及他对面向对象技术的实践。William Tepfenhart提供了很多优秀的“处方”，并且组织了材料，使得开发者可以快速地成为开发小组中有贡献的成员。

最后，我们对提供宝贵反馈意见的审阅者表示感谢和感激，他们是Rex Jaeschke（ANCI C委员会的主席，独立的顾问和作者）和Robert Taylor（Taylor计算公司）。作为作者，我们十分高兴接受所有有关错误、疏忽、不准确、不真实、思想模糊以及有关质量的批评。我们欢迎所有建设性的评论，并将愉快地忽略破坏性的评论。

Richard C.Lee

William M.Tepfenhart

# 目　录