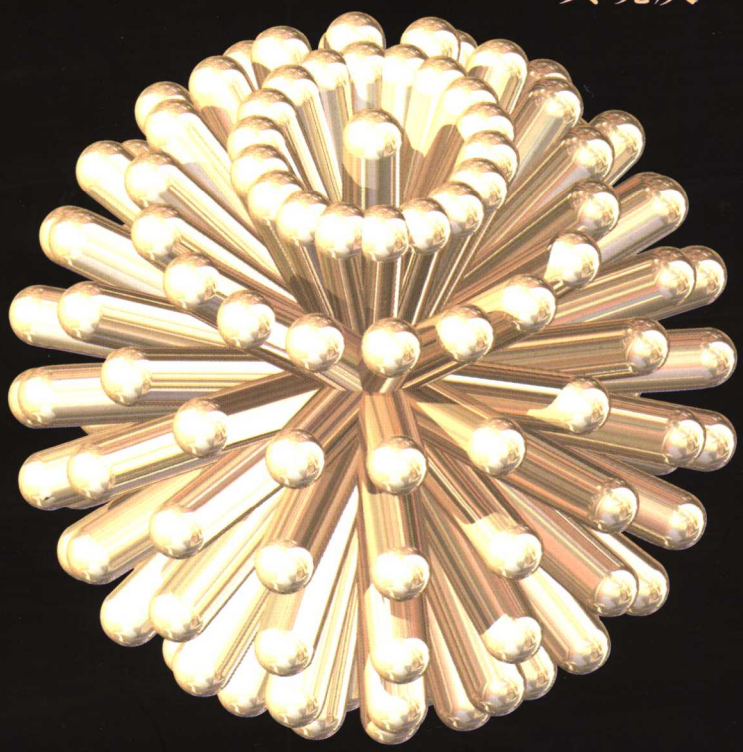
Mc Graw Hill Education

世界著名计算机教材精选

面向对象设计 UML实践 第2版

Mark Priestley 著
龚晓庆 卞 雷 等译
王少锋 审



PRACTICAL OBJECT-ORIENTED DESIGN WITH UML

Second Edition

清华大学出版社

世界著名计算机教材精选

面向对象设计UML实践

（第2版）

Mark Priestley 著
龚晓庆 卞 雷 等译
王少锋 审

清华大学出版社
北 京

Mark Priestley
Practical Object-Oriented Design with UML, Second Edition
EISBN: 0-07-710393-9

图书在版编目(CIP)数据

面向对象设计 UML 实践（第 2 版）/普里斯特（Priestley, M.）著；龚晓庆，卞雷等译. —北京：清华大学出版社，2005.5
（世界著名计算机教材精选）
书名原文：Practical Object-Oriented Design with UML
ISBN 7-302-10587-1

Ⅰ. 面…　Ⅱ. ①普…　②龚…　③卞…　Ⅲ. 面向对象语言，UML－程序设计－高等学校－教材　Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2005）第 015590 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
社 总 机：010-62770175　　**客户服务**：010-62776969
责任编辑：龙启铭
封面设计：常雪影
印 刷 者：北京密云胶印厂
装 订 者：三河市金元装订厂
发 行 者：新华书店总店北京发行所
开　　本：185×260　**印张**：19.25　**字数**：451 千字
版　　次：2005 年 5 月第 1 版　2005 年 5 月第 1 次印刷
书　　号：ISBN 7-302-10587-1/TP・7180
印　　数：1～4000
定　　价：39.00 元

序　言[1]

Mr Palomer 的规则一直在逐渐改变：现在，在综合过程中他需要多种多样的模型，很可能是可以互换的，以便找出一个最符合于实际情况的模型，而这个实际情况就其本身来说总是由许多在时间上和空间上不同的实际情况构成。

Italo Calvino

本书的目的是提供一本实用的和易于理解的面向对象设计的入门教程，它要求读者具有面向对象程序设计语言的知识（最好是 Java），而且阐述了 UML 的原则和应用。本书针对的读者主要是计算机科学或软件工程专业的大学本科高年级学生或硕士生，当然也希望其他读者会发觉这本书是有用的。

本书的总体设想是强调设计的表示法与代码之间的联系。现在已经有许多讨论用 UML 进行系统分析和设计的教程，不过对最终产品，即被开发系统的代码，给予特别注意的还不多见。然而 UML 实质上是一种表达面向对象程序的设计语言，从这个角度考虑该语言的表示法和语义看来是很自然的。在过去几年，我已发现这是把设计表示法的真实含义传授给学生的好方法。

与这个总体设想有关，本书有两个主要目标。第一个是提供一个使用 UML 描述的面向对象开发的完整示例。开始讲述需求，最后讲述一个完整的可执行代码，这个代码可以运行、修改和扩展。

教程的目标限制了可以考虑的事例的规模。为此，本书采用一个典型的独立的桌面应用系统，作为范例体系结构。它支持图形用户界面并与关系数据库接口。在这个框架中，正文仔细地对某些核心功能的开发进行了探讨，并把系统的扩展作为练习留给读者。

第二个目标是对 UML 在开发这种应用的重要方面提供指导。特别把重点放在清楚地阐明此设计语言的结构和表示法，并通过示例说明设计和面向对象程序实现之间的紧密关系。这些问题在许多书中讲的相当粗略。然而，如果对此没有清楚的理解，就难以正确使用 UML。

UML 是一个庞大而复杂的语言，学习 UML 时的危险是被这些表示法的细节所淹没。为了避免这点，本书使用的是足以用于开发桌面应用的 UML 的一个子集。并发性、活动图和除了部署图的简要叙述外的其他部分，都作了相当大的省略。UML 语言的这些方面对于 UML 的“工业化”应用显然是重要的，但是这些已超出本书针对的读者的经验。

本书的结构

第 1 章导论之后，第 2 章结合一个简单的程序设计示例介绍对象建模的基本概念。第 3~7 章是使用 UML 的一个案例研究，而第 8~12 章系统地介绍最重要的 UML 表示法。这两部分是相互独立的，可以按照如图 0.1 所示的不同阅读计划学习。第 13 章讨论实现 UML

[1] 本书英文影印版已由清华大学出版社出版，书号为 7-302-08784-9，定价为 39.00 元。

的策略，第 14 章总体讨论了面向对象设计的一些基本原则。

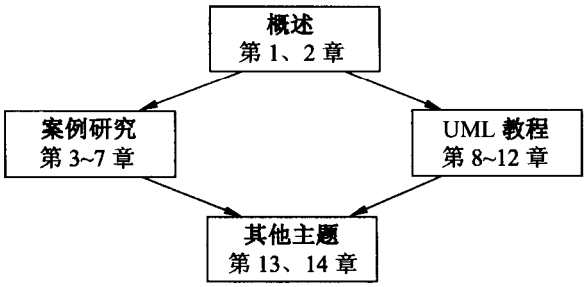


图 0.1 各章之间的关系

第 2 版的改变

第 2 版最大的改变是用一个新的简单餐馆预约系统案例，替换了图编辑器示例。这就提供了一个比图编辑器具有更“贴近生活”环境的应用，许多学生发现这个案例与他们更接近。与原来的案例相比，这个案例也允许更自然地介绍不同框架层的概念。容易看到，现在的第 4~7 章包括了这些题目。

虽然本书的重点是语言而不是过程，然而在任何有实用价值的方式中，使这两部分完全分离是不可能的。在本书新的第 3 章中，现在包括了软件开发过程中一些问题的明确讨论，并对统一过程给出了一个简要说明。

其余各章相当多的内容与第 1 版是一样的，只是在内容上和表达上有少量修改。为了给新的章节和案例让出地方，第 1 版中的某些资料已从这一版中删去，显而易见的是第 2 个案例。所有已删去的资料，包括图编辑器的示例，仍然可以从本书的网站得到。

其 他 资 料

本书的 Web 页提供访问本书使用的示例的源代码、所有练习的答案和第 1 版的资料。这些资料可以在下述 URL 找到：

http://www.mcgraw-hill.co.uk/textbooks/priestley

教师手册、幻灯片、本书中使用的图和增加的练习，可以供真正使用本书进行课堂教学的大学教师使用。如何获得手册中的这些信息可以在出版商的网站上找到。

致 谢

我非常感激在本书新版的准备中使用过早期版本的学生和自始至终参加这个餐馆预定系统早期描述的学生。我还要感谢 Michael Richards，是他最早提出了这个案例的思想。

Mark Priestley

目　录

第1章 UML导论

统一建模语言（Unified Modeling Language），简称UML，按照UML的设计者所言，是一种“通用的可视建模语言，用于说明、可视化、构造并文档化软件系统的体系结构”。本章阐述软件开发过程中如何使用模型，以及像UML这种语言的作用。文中描述了UML的高级结构及其语义的非形式说明，以及设计表示法和代码之间的关系。

1.1 模型与建模

模型在软件开发中的使用非常普遍。本节先介绍模型的两种典型用法，即在描述现实世界的应用中和实现应用的软件系统中的用法，随后讨论这两种模型之间的关系。

1.1.1 软件模型

软件开发通常按以下的方式进行：一旦决定建立一个新的系统，就要写一个非正式的描述说明软件应该做什么，这个描述被称为**需求说明书**（requirements specification），通常是经过与系统未来的用户磋商制定的，并且可以作为用户和软件供应商之间的正式合同的基础。

完成后的需求说明书移交给负责编写软件的程序员或者项目组，他们相对孤立地根据说明书编写程序。幸运的话，结果程序能够按时完成，不超出预算，而且能够满足最初方案目标用户的需要。但不幸的是在许多情况下，事情并不是这样。

许多软件项目的失败引发了人们对软件开发方法的研究，试图了解项目为何失败，结果得到了许多对如何改进软件开发过程的建议。这些建议通常以**过程模型**的形式，描述了开发所涉及的多个活动及其应该执行的次序。

过程模型可以用图解的形式表示。例如，图1.1表示一个非常简单的过程，其中直接从系统需求开始编写代码，没有中间步骤。图中除了圆角矩形表示的过程之外，还显示了过程中每个阶段的产物。如果过程中的两个阶段依次进行，一个阶段的输出通常就作为下一个阶段的输入，如虚线箭头所示。

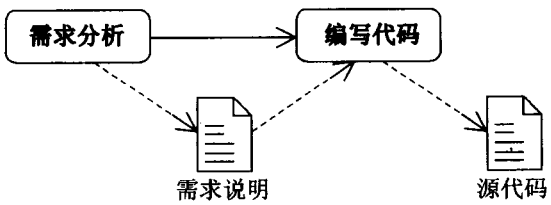


图1.1 软件开发的基本模型

开发初期产生的需求说明书可以采取多种形式。书面的说明书可以是所需系统的非常不正规的概要轮廓，也可以是非常详细、井井有条的功能描述。在小规模的开发中，最初的系统描述甚至可能不会写下来，而只是程序员对需要什么的非正式的理解。在有些情况下，可能会和未来的用户合作开发一个原型系统，成为后续开发工作的基础。上面所述的所有可能性都包括在“需求说明书”这个一般术语中，但并不意味着只有书面的文档才能够作为后继开发工作的起点。

还要注意的是，图 1.1 没有描述整个软件生命周期。在本书中，术语“软件开发”是在比较狭隘的意义上使用的，它只包括软件系统的设计和实现，而忽略了生命周期的其他一些重要组成部分。一个完整的项目计划还应该提供如项目管理、需求分析、质量保证和维护等关键活动。

单个程序员在编写简单的小程序时几乎不需要比图 1.1 更多的组织开发过程。有经验的程序员在写程序时心中会很清楚程序的数据和子程序结构，如果程序的行为不是预期的那样，他们能够直接对代码进行必要的修改。在某些情况下，这是完全适宜的工作方式。

然而，对比较大的程序，尤其是如果不止一个人参与开发时，在过程中引入更多的结构通常是必要的。软件开发不再被看作是单独的自由的活动，而是分割为多个子任务，每个子任务一般都涉及一些中间文档资料的产生。

图 1.2 描述的是一个比图 1.1 稍微复杂一些的软件开发过程。在这种情况下，程序员不再只是根据需求说明书编写代码，而是先创建一个结构图，用以表示程序的总体功能如何划分为一些模块或子程序，并说明这些子程序之间的调用关系。

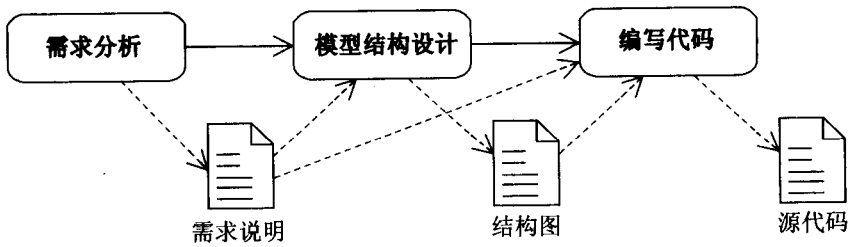


图 1.2 更复杂的软件开发过程

这个过程模型表明，结构图以需求说明书中包含的信息为基础，说明书和结构图在编写最终代码时都要使用。程序员可以使用结构图使程序的总体结构清楚明确，并在编写各个子过程的代码时参考说明书核对所需功能的详细说明。

在开发一个软件期间所产生的中间描述或文档称为**模型**。图 1.2 中给出的结构图在此意义上即是模型的一个示例。模型展现系统的一个抽象视图，突出了系统设计的某些重要方面，如子程序和它们的关系，而忽略了大量的底层细节，如各个子程序代码的编写。因此，模型比系统的全部代码更容易理解，通常用来阐明系统的整体结构或体系结构。上面结构图中所包含的子程序调用结构就是这种结构的一个示例。

随着开发的系统规模更大、更复杂以及开发组人数的增加，需要在过程中引入更多的规定。这种复杂性的增加的一个外部表现就是在开发期间使用了更广泛的模型。实际上，软件设计有时就定义为构造一系列模型，这些模型越来越详细地描述系统的重要方面，直

到获得对需求的充分理解，能够开始编程为止。

因此，使用模型是软件设计的中心，它具有两个重要的优点，有助于处理重大软件开发中的复杂性。第一，系统要作为整体来理解可能过于复杂，模型则提供了对系统重要方面的简明描述；第二，模型为开发组的不同成员之间以及开发组和外界如客户之间提供了一种颇有价值的通信手段。本书描述面向对象设计中所用的模型，并举了一些模型的应用实例。

1.1.2　应用模型

在软件开发进入系统设计和编码阶段之前，也用模型来帮助理解系统所针对的应用领域。这些模型通常称为**分析模型**，相对应的是**设计模型**，如上面所讨论的结构图。这两类模型可以通过这样的事实区分：分析模型不同于设计模型，它不涉及要开发的系统的任何特性，而是力求捕捉“现实世界”中的业务的某些方面和特性。

总之，分析模型和设计模型满足相同的需要并带来同样的益处。它们支持的或与之相互作用的软件系统和现实世界系统往往都非常复杂，千头万绪。为了管理这种复杂性，系统的描述需要着重于结构而非细节，并要提供系统的一个抽象视图。这个抽象视图确切的特性将依赖它产生的目的，而且通常需要多个这样的视图或模型为系统提供一个足够的全景。

通常，分析模型描述应用中处理的数据和处理数据的各种过程。在传统的分析方法中，这些模型用图表示，如逻辑数据模型和数据流图。值得注意的是，使用分析模型描述业务过程，早于并且独立于这种过程的计算机化，例如，组织结构图和说明特定生产过程的示意图在商业和工业中已经使用了相当长的时间。

1.1.3　分析模型和设计模型的关系

在开发任何有效的软件系统期间，上面定义的分析模型和设计模型很可能都要产生。这就引出了一个问题：它们之间的关系是怎样的？

软件开发过程传统上划分为若干阶段。分析阶段最终以产生一组分析模型而结束，随后是设计阶段，它产生一组设计模型。在这种情况下，分析模型用来形成设计阶段的输入，设计阶段的任务是创建支持分析模型中规定的特性和要求的结构。

这样划分工作有一个问题，在多数情况下，分析和设计模型产物中使用的是完全不同的语言和表示法，这就导致从一个阶段转移到下一个阶段时需要一个翻译过程，分析模型中包含的信息必须用设计模型要求的表示法重新阐述。

显然，这里存在一个危险，就是这个过程容易出错，而且很浪费。问题是，如果在开发过程中剩余的阶段要用设计模型取代分析模型，那么为什么还要特意创建一个分析模型呢？而且，如果两种模型之间存在表示法上的差异，就难以肯定分析模型中所包含的全部信息都准确地被提取并且用设计表示法表示。

面向对象技术的一个承诺就是，通过对分析和设计使用同样的模型和建模概念，来消除这些问题。按照这种设想，分析和设计模型之间任何明显的差别都将会消除。显然，设