国外计算机科学经典教材

Object-Oriented Analysis & Design

Understanding System Development with UML 2.0

面向对象分析与设计

(UML 2.0 版)

(美) Mike O'Docherty 著
俞志翔 译

清华大学出版社

国外计算机科学经典教材

面向对象分析与设计

(UML 2.0 版)

(美) Mike O'Docherty 著

俞志翔 译

清华大学出版社

北 京

Mike O'Docherty
Object-Oriented Analysis and Design: Understanding System Development with UML 2.0
EISBN：0-470-09240-8

图书在版编目(CIP)数据

面向对象分析与设计(UML 2.0 版)/(美)多切蒂(O'Docherty,M.)著；俞志翔 译. —北京：清华大学出版社，2006.4
书名原文：Object-Oriented Analysis and Design:Understanding System Development with UML 2.0
(国外计算机科学经典教材)
ISBN 7-302-12546-5

Ⅰ. 面… Ⅱ. ①多… ②俞… 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(2006)第 009825 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
组稿编辑：曹　康
文稿编辑：于　平
封面设计：康　博
版式设计：康　博
印 刷 者：北京市人民文学印刷厂
装 订 者：三河市金元印装有限公司
发 行 者：新华书店总店北京发行所
开　　本：185×260　**印张**：23.5　**字数**：601 千字
版　　次：2006 年 4 月第 1 版　2006 年 4 月第1 次印刷
书　　号：ISBN 7-302-12546-5/TP・8025
印　　数：1～4000
定　　价：42.00 元

出 版 说 明

近年来，我国高等学校的计算机学科教育进行了较大的改革，急需一批门类齐全、具有国际水平的计算机经典教材，以适应当前的教学需要。引进国外经典教材，可以了解并吸收国际先进的教学思想和教学方法，使我国的计算机学科教育能够与国际接轨，从而培育更多具有国际水准的计算机专业人才，增强我国信息产业的核心竞争力。Pearson、Thomson、McGraw-Hill、Springer、John Wiley 等出版集团都是全球最有影响的图书出版机构，它们在高等教育领域也都有着不凡的表现，为全世界的高等学校计算机教学提供了大量的优秀教材。为了满足我国高等学校计算机学科的教学需要，我社计划从这些知名的国外出版集团引进计算机学科经典教材。

为了保证引进版教材的质量，我们在全国范围内组织并成立了“清华大学计算机外版教材编审委员会”（以下简称“编委会”），旨在对引进教材进行审定、对教材翻译质量进行评审。“编委会”成员皆为全国各类重点院校教学与科研第一线的知名教授，其中许多教授为各校相关院、系的院长或系主任。“编委会”一致认为，引进版教材要能够满足国内各高校计算机教学与国际接轨的需要，要有特色风格，有创新性、先进性、示范性和一定的前瞻性，是真正的经典教材。为了保证外版教材的翻译质量，我们聘请了高校计算机相关专业教学与科研第一线的教师及相关领域的专家担纲译者，其中许多译者为海外留学回国人员。为了尽可能地保留与发扬教材原著的精华，在经过翻译和编辑加工之后，由“编委会”成员对文稿进行审定，以最大程度地弥补和修正在前面一系列加工过程中对教材造成的误差和瑕疵。

由于时间紧迫和能力所限，本套外版教材在出版过程中还可能存在一些不足和遗憾，欢迎广大师生批评指正。同时，也欢迎读者朋友积极向我们推荐各类优秀的国外计算机教材，共同为我国高等学校的计算机教育事业贡献力量。

清华大学出版社

国外计算机科学经典教材

编审委员会

目　录

第1章 入 门

本书的目的是让读者基本掌握面向对象的软件开发所使用的过程和技术，并学会使用面向对象的技术编写计算机程序。统一建模语言(Unified Modeling Language，UML)是软件业的标准语言。

不管您是一位在校大学生，或是正在接受商业培训的工作人员，或是一位打算转向面向对象技术的、经验丰富的软件开发人员。无论如何，本书都会对您有所帮助。读者不需要有什么背景知识，本书也不打算讲述有关面向对象的所有知识，而是介绍该过程的基本组成部分，使您能更高效地完成工作。

本书的内容覆盖面比较广，但并不深邃，就到教会您编写代码为止。对如何编写代码的描述实际上就是选择特定的编程语言，为您选择合适的语言做准备。本书是掌握任何纯粹的、面向对象的编程语言的一门前导课程。

第一章主要讲述了本书的背景，概述了本书的内容和学习本书的方式。

1.1 背景

近来，新软件通常都是面向对象的。也就是说，使用称为“对象(object)”的抽象物体来编写软件。自然，商业软件开发要比简单地编写几行代码复杂得多：它需要调研商务需求、分析问题、设计解决方案等。在开发的每个阶段都应使用对象，因为对象减少了必须理解的信息量，加强了开发小组成员之间的交流。

1.2 编程简史

商业编程有许多发展阶段，“面向对象”是最新的一代：

- 机器码：编程使用二进制数字。
- 汇编语言：编程使用字母数字符号作为机器码的速记方式。汇编语言通过汇编程序(assembler)转换为机器码。
- 高级语言：编程使用有高级结构(如类型、函数、循环和分支)的语言(例如 Fortran 和 COBOL)。高级语言(和以后的各种编程语言)使用编译程序转换为机器码。
- 结构化编程：编程使用更简洁的高级语言(例如 Pascal、Modula 和 Ada)，它们的特点是程序员犯的错误更少，程序分解为子任务和子系统的方式更规范。
- 面向对象编程：编程使用数据和函数的独立模块，这些模块对应于问题域中的概念，例如 Customer 或 ScrollBar。这种模块化使程序员的错误更少，并允许在不同的程序中重复使用代码。优秀的面向对象编程语言有 Java 和 Eiffel，因为它们的设计非常简洁，

很纯粹，还可以移植(可以在许多平台上使用)。其他语言有 Smalltalk、C#，以及最初是结构化语言、后来包含了面向对象扩展的任何语言(C++和 Pascal 的各种版本)。

读者可能听说过函数化编程(functional programming)和逻辑编程(logic programming)，目前它们已退出了市场。

目前，这些编程语言或多或少都在使用。选择使用哪种语言取决于使用场合、个人喜好和要解决的问题，例如，视频游戏需要的是速度，所以有时用汇编语言编写。

1.3 方法学

20 世纪 80 年代，结构化编程语言开始流行，有经验的程序员开始描述如何控制整个软件开发过程，从任务陈述开始，一直到对已完成产品的维护，都要进行详细的描述。于是诞生了结构化方法学，例如 SSADM [Weaver 等 02]。方法学描述了开发小组编写高质量系统所应遵循的步骤，规定了产品应包含什么内容(文档、图表、代码等)，产品应采用什么形式(如目录、图标、编码样式)。

90 年代，面向对象编程开始盛行，开发人员又发明了面向对象方法学，更适合进行面向对象的编程。早期的面向对象方法学有 Booch 方法[Booch 93]、Objectory[Jacobson 等，92]和 OMT [Rumbaugh 等，91]。近来，市场上的主导方法学是 IBM 提出的 Rational 统一过程(Rational Unified Process，RUP)，它归 IBM(www.rational.com)所有。大致说来，RUP 是 Objectory、Booch 和 OMT 的综合。另一个日益流行的方法学是极限编程(extreme programming，XP)[Beck 99]，即所谓的“敏捷”方法学，在软件开发业中，敏捷意味着反映软件需求的变化或对问题的理解的变化。

本书使用的方法学是 Ripple(详见附录 A)，这是一种简化的方法学，基于被广泛接受的理论和实践。因此，读者不必学习完整方法学的复杂内容，本书也不会明确地告诉读者在每个阶段应做什么，而是允许读者自由发挥，推陈出新。

1.4 关于本书

为了避免混淆，本书不对结构化方法学和面向对象方法学进行详细的比较，而直接介绍面向对象的软件开发，就好像传统方法从来不存在一样。我们将讲述编写面向对象的大型软件时所需要掌握的所有内容(读者只需努力学习，积累经验即可)。

您不会用到以前已很熟悉的结构化技术，但不必担心，面向对象方法确实很有效，自从 70 年代开始人们就在研究它，在 90 年代面市以来，每天都有上百万的开发人员在使用它。

与以前一样，软件开发的目的是编码、编码、再编码。无论您的背景是什么，只要有编程经验就行。如果没有编程经验，学习本书是比较直截了当的方法——本书不会让您为了晦涩的术语和难以理解的技巧而头痛。但在开始之前，应熟悉基本的计算机概念，例如硬件、软件和网络。最起码要用高级语言编写过几百行代码。

1.4.1 内容概述

本书并没有介绍编写代码的所有细节，但常常需要使用代码段来演示。本书所有的代码段都是用 Java[Joy 等，00]编写的，因为 Java 非常普及、纯粹、简单，也是免费的。每个代码段的含义都会在文中详细解释。如果您不喜欢使用 Java，这些代码段还能很容易地转换为其他语言，例如 C#，因为本书的代码没有使用 Java 的独特元素。用 Java 完成的所有任务都可以用其他任何一种语言完成。同样，系统设计的讨论也集中于 Java 技术，而不是.NET 技术，因为 Java 和.NET 支持类似的功能，选择 Java 还是.NET 完全是个人喜好。Java 系统设计的所有内容都可以以相同的方式用.NET 实现。

但要注意，为了便于演示，本书描述的类(例如 Iterator 和 List)并不匹配 Java 库中的类。使用本书肯定可以理解 Java 的工作方式，熟悉大多数语法，但本书不能替代单纯介绍 Java 语言的图书[Campione 等，00]。所以，在编写 Java 代码时，最好手中有一本有关 Java 库的书，因为每个人都不可能记住上千个类的所有细节。

另一个难点是项目管理(规划、调度等问题)。本书没有讨论项目管理，因为我们主要论述的是技术问题，而不是人的因素。

在演示过程中使用的表示法称为统一建模语言(UML)[OMG 03a]，它是已被广泛接受的软件图标准。在 UML 的约定中，一些线条比另外一些粗，一些字符采用黑体或斜体。尽管只有斜体的内容比较重要，但这些约定很难用手工绘制出来(在纸或白板上)，所以在手工绘制时可以忽略其他约定。对于斜体的内容，还可以在适当的时候用其他方式替代。

UML 不能满足所有文档的需要，所以本书的一些文档取自 RUP。

1.4.2 案例分析

本书使用的案例分析是一个租用和预约系统，称为 Coot，由一家虚拟公司 Nowhere Cars 开发。因此，书中的许多例子都以某种方式使用汽车。大多数例子使用相同的应用程序域，在阅读本书时就不必在不同的域中来回转换。为了简单起见，大多数讨论都会提及 Coot 中给客户提供 Internet 功能的部门，这个简化的系统称为 iCoot。

由于有许多概念要解释，而且并不是每个概念都与汽车的租赁相关，所以本书的一些例子并不涉及 iCoot(它们更适合汽车销售员或汽车结构)。但是，每个重要的图表都取自完整的系统。

面向对象的新手常常希望有对案例的完整分析，所以附录 B 包含了 iCoot 的完整说明，供读者进一步学习。iCoot 文档虽然是一个真实有效的软件，但很容易理解。

如果读者希望像学生一样实践本书描述的技术，可以在 www.wiley.com/go/odocherty 上找到一组 AQS(Automated Quiz System，自动组卷系统)练习题。AQS 是一个提取多项选择题的在线工具。这些练习题按照本书的主要章节来安排，读者在阅读的过程中就可以完成它们。教师如果想把本书用作大学教材或商务培训教材，在注册后就可以获得 AQS 练习题的答案。

1.4.3 导航

在阅读完第 1 章后，就可以用很直接的方式按顺序导航和通读所有的章节了。

或者，如果已经熟悉面向对象的概念和术语，就可以跳过第 2、3 和 4 章，直接阅读第 5 章。如果对面向对象领域非常陌生，就应先阅读第 2 章，但可以把第 3 和 4 章放在后面。

如果对 Ripple 的概述和方法学不感兴趣，而是喜欢直接进入细节，就可以跳过第 5 章，直接阅读第 6 章。

其余章节的结构比较严谨，是典型的软件开发过程，所以不应跳读。第 11 章最好当作主要内容来阅读，但由于其中的许多内容比较高级，也可以留到以后学习。

附录 A 可以用作自我测验，也可以在尝试案例分析时用作参考。

附录 B 包含 iCoot 案例分析的所有内容，其顺序按照一般开发过程来组织。使用这些内容和主要的开发章节，就可以了解 iCoot 系统的开发过程。附录 B 还包含 iCoot 项目的术语表，在 iCoot 的开发过程中，这个术语表会不断地更新。使用它可以了解一般术语表的内容，和查找 iCoot 术语定义的方式。

如果需要学习如何绘制 UML 图，就可以参考附录 C。

第 I 部分

设 置 场 景

第 2 章　对象的概念

本章介绍的概念都来自于面向对象编程(object-oriented programming)。一般说来，编程语言都是在帮助使用它们的方法学之前发明的，面向对象的概念非常有意思，因为面向对象的开发允许用真实世界中的术语思考问题，而不是局限于计算机的需要。

学习目标：

- 理解软件对象的含义
- 理解对象使用消息进行交流以完成任务的方式
- 理解不再需要某对象时会发生什么(垃圾收集)
- 理解类的含义
- 理解重用代码的方式

2.1　引言

面向对象(做事的方式)的基本概念相当容易理解和应用。Alan Kay 是 Smalltalk 的发明者，他早在 1968 年开始就为“所有年龄段的孩子的个人计算机”[Kay 72]工作。他的目标是孩子，所以基本概念非常简单。

为什么所有的事情都与对象有关？开发人员没有更好的理由改变软件开发的根本吗？在早期阶段，使用对象的一些理由看起来相当模糊，尤其是对以前的技术(结构化编程和结构化方法学)没有什么经验时，这些理由就更难理解了。出现(或诞生)面向对象方法，是因为人们发现，在指定的时间和预算内生产出高质量的系统越来越难了，尤其是涉及许多人的大型系统。

一旦阅读完本书，就会觉得下面给出的理由很有道理，而且会对大部分理由持赞同态度。这里只列出了面向对象的一般理由：

- 对象更便于人们理解：这是因为对象派生自我们试图自动化的业务，而不是派生自受计算机上的过程或数据存储需求过早影响的业务。例如，在银行系统中，是根据银行账户、银行出纳员和顾客来编程，而不是直接深入账户记录、取款和存款过程、贷款资格的算法等。
- 专业人员可以更好地交流。随着时间的推移，软件业已经形成了职业阶梯，新人可以积累知识和经验，一步步向上爬。一般，第一步是程序员：修改其他人编写的代码中的错误。第二步是高级程序员：自己编写代码。第三步是设计师：根据需要决定编写什么代码，最后一步是分析师：与客户交流，了解他们的需求，然后总结出最终完成的系统必须能做的工作。

这样的职业阶梯本身并不是什么坏事。但每位专业人士都希望学习一套全新的概念和技