国外计算机科学教材系列

实用面向对象软件工程教程

case studies in

OBJECT ORIENTED ANALYSIS & DESIGN

Edward Yourdon & Carl Argila 著

殷人昆 田金兰 马晓勤 译

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

PRENTICE HALL 出版公司

国外计算机科学教材系列

实用面向对象软件工程教程

case studies in
OBJECT ORIENTED ANALYSIS & DESIGN

Edward Yourdon & Carl Argila 著

殷人昆　田金兰　马晓勤　译

PRENTICE HALL 出版公司

電子工業出版社

内 容 提 要

本书是一本有关面向对象软件开发的事例分析的教材。它给出两个典型的事例分析：一个是 CAD 类型的软件，一个是 MIS 类型的软件。从问题的提出，到面向对象的分析、设计、实现，都给出了很好的指导。特别是在分析时给出的三视图模型，即实体-关系模型、数据流模型和状态-迁移模型，以及在识别对象时用到的基于语言的信息流分析等，都为读者提供了有力的工具。此外，它在介绍事例时把软件工程和面向对象的基本概念也作了介绍。因此，本书对于学习软件工程和面向对象技术的读者是一本很好的引路教材。

本书可以作为计算机系软件工程课程的辅助教材，并作为非计算机专业学生和研究生的软件工程课程的主要教材。对于有兴趣使用面向对象的软件开发人员，也可作为参考书。

丛 书 名： 国外计算机科学教材系列
原 书 名： case studies in OBJECT ORIENTED ANALYSIS & DESIGN
书　　名： **实用面向对象软件工程教程**
著　　者： Edward Yourdon & Carl Argila
译　　者： 殷人昆　田金兰　马晓勤
责任编辑： 陆伯雄
印 刷 者： 顺义县天竺颖华印刷厂印刷
出版发行：电子工业出版社出版、发行　URL：http://www.phei.com.cn
　　北京市海淀区万寿路 173 信箱　邮编　100036　发行部电话 68214070
经　　销：各地新华书店经销
开　　本：787×1092 1/16　印张：19　字数：422 千字
版　　次：1998 年 6 月第 1 版　1998 年 6 月第 1 次印刷
定　　价：30.00 元
印　　数：7000 册
书　　号：ISBN 7-5053-4603-2/TP·2184
著作权合同登记号　图字：01-97-1864

出版说明

计算机科学的迅速发展是20世纪科学发展史上最伟大的事件之一。从1946年第一台笨重而体积庞大的计算机的发明至今，仅仅半个多世纪，计算机已经变得小巧无比却又能力非凡。它的应用已经渗透到了社会的各个方面，成为当今所谓的信息社会的最显著的特征。

处于世纪之交科技进步的大潮中，我国正在加强计算机科学的高等教育，着眼于为下一世纪培养高素质的计算机人才，以适应信息社会加速度发展的需要。当前，全国各类高等院校已经或计划在各专业基础课程规划中增加计算机科学的课程内容，而作为与计算机科学密切相关的计算机、通信、信息等专业，更是在酝酿着教学的全面革新，以期规划出一整套面向21世纪的、具有中国高校计算机教育特色的课程计划和教材体系。值此，我们不妨借鉴并引进国外具有先进性、实用性和权威性的大学计算机教材，洋为中用，以更好地服务于国内的高校教育。

美国Prentice Hall出版公司是享誉世界的高校教材出版商，自1913年公司成立以来，即致力于教育图书的出版。它所出版的计算机教材在美国为众多大学所采用，其中有不少是专业领域中的经典名著。许多蜚声世界的教授学者成为该公司的资深作者，如：道格拉斯·科默(Douglas Comer)，安德鲁·坦尼伯姆(Andrew Tanenbaum)，威廉·斯大林(William Stallings)……几十年来，他们的著作教育了一批批不同肤色的莘莘学子，使这些教材同时也成为全人类的共同财富。

为了保证本系列教材翻译出版的质量，电子工业出版社和Prentice Hall出版公司共同约请北京地区的清华大学、北京大学、北京航空航天大学，上海地区的上海交通大学、复旦大学，南京地区的南京大学、解放军通信工程学院等全国著名的高等院校的教学第一线的几十位教师参加翻译工作。这中间有正在讲授同类教材的年轻教师和博士，有积累了几十年教学经验的教授和博士生导师，还有我国著名的计算机科学家。他们的辛勤劳动保证了本系列丛书得以高质量地出版面世。

如此大规模地引进计算机科学系列教材，在我们还是第一次。除缺乏经验之外，还由于我们对计算机科学的发展，对中国高校计算机教育特点认识的不足，致使在选题确定、翻译、出版等工作中，肯定存在许多遗憾和不足之处，恳请广大师生和其他读者提出批评、建议。

电子工业出版社
URL：http://www.phei.com.cn
Prentice Hall出版公司
URL：http://www.prenhall.com

译者的话

从事软件开发的人们常常有这样的体会：在软件开发的过程中，使用者或用户会不断地提出各种各样的更改要求，使得软件开发人员不得不对软件进行修改。这将导致软件开发的进度一拖再拖，软件开发的成本不断增加。此外，在软件投入使用后，为了排除在开发过程中遗留下来的错误或缺陷，为了改进软件的性能、增强软件的功能，为了能够适应不断出现的新机器、新操作系统或新数据环境，都需要修改软件。这些修改或大或小，大者甚至将对软件作重大的“外科手术”。此时，万一软件人员计划不周或考虑不细，不但老的错误没有彻底纠正，还会引入新的错误，导致软件质量下降，软件使用寿命缩短。

面向对象技术能够减轻软件修改的困难。使用面向对象技术开发出来的软件结构是建立在现实世界的实体或对象的基础上的，它把软件的功能分散到各个对象中间。使用者和用户提出的修改要求大多是功能上的，所面对的对象基本不动。对于使用面向对象技术开发出来的软件来说，软件的修改主要集中于封装在软件对象内部的属性和服务上，只要对象界面不动，整个软件的体系结构可以不动。这种修改的局部化保持了软件结构的稳定性，使得在修改过程中引入新错误的可能性达到最小，同时也减轻了软件修改的工作量和难度。

另一方面，软件的质量主要指软件产品的质量和软件工程过程的质量。遗憾的是，许多从事软件开发的人把眼光只盯在软件产品的质量上，忽视软件开发过程的质量，觉得按软件工程的要求做太麻烦，既没有时间也没有必要，只要把程序编好，能够实现用户的要求就可以了。其结果是，他们开发的软件模块结构不合理，模块之间界面复杂，没有考虑将来的修改而使得程序缺乏灵活性。这将导致以后的修改和扩充困难，每次修改都要对软件的结构进行变动，对程序的内部代码进行改变，给软件的开发和维护造成障碍。

现有的软件工程教科书对传统的面向过程的软件开发过程论述十分详细，但对面向对象软件的开发过程言之泛泛，以至于许多学生在学习过后还不知道该如何做。他们非常希望能够了解一个具体的面向对象软件的开发全貌。本书是一本有关面向对象软件开发的事例分析的教材。它给出两个典型的事例分析：一个是 CAD 类型的软件，一个是 MIS 类型的软件。从问题的提出，到面向对象的分析、设计、实现，都给出了很好的指导。特别是在分析时给出的三视图模型，即实体-关系模型、数据流模型和状态-迁移模型，以及在识别对象时用到的基于语言的信息流分析等，都为读者提供了有力的工具。此外，它在介绍事例时把软件工程和面向对象的基本概念也作了介绍。因此，我们认为本书对于学习软件工程和面向对象技术的读者是一本很好的引路教材。我们在为北京图书馆和北京邮电局开发软件时，曾经仿照本书提供的方法，取得了很好的效果。

本书所介绍的是 Coad & Yourdon 的面向对象分析和设计技术，目前国内外已有许多软件工具支持这个技术。例如，北京大学开发的青鸟软件开发环境。此外，还有许多面向对象的开发方法，例如 Rumbaugh 等人的 OMT 技术、Booch 技术等。事实上，各种面向对象的方法都在相互借鉴，不断补充与发展。学会一种方法，就可以举一反三，事半功倍。

本书可以作为计算机系软件工程课程的辅助教材，并作为非计算机专业学生和研究生的软件工程课程的主要教材。对于有兴趣使用面向对象的软件开发人员，也可作为参考书。

译者

1998 年 3 月 2 日於北京清华园

前　言

作为一个数学专业的研究生就不得不忍受那些无穷尽的数学定理。每个定理都经过那些博学的教授的严格证明。而每一个证明作为最后的结果都显得那样的完美和谐。但作为一个学生却觉得这些证明并不是那样的完美和谐，而是那样的晦涩难懂。

有的时候，当证明很长时，教授就会显得有些困窘，不太确信下一步应该是什么。他会带着一种沉思的表情，走到黑板边上，拿起一支粉笔，开始画起图。他画着一些圆、弧、直线，直到最后他的脸上出现一种若有所思的神情。他迅速地擦掉他所画的那些东西，走到教室前，继续他的证明。但你可以想象得出，学生们肯定对教授擦去的东西更感兴趣，而不是教授所要证明的东西。

对一个庞大的、复杂的软件系统的分析和设计有些类似于数学证明，它们总是作为一个成品出现。它们会被大量的文档加以说明，或作为一个最终的模型。但很少有软件分析人员或设计人员记录他们的分析或设计过程。在以前，一个精明老练的软件管理人员会偷偷进入工作人员的工作环境，查看他们的废物箱，从而找出他们的思路。然而随着 CASE 工具的出现，这种事情就越来越不可能发生了。而且，几乎没有什么软件分析和设计方面的书能很深入地涉及到启发式的方法。

这种状况在面向对象分析和设计领域中尤为厉害。虽然有关这方面的书有很多(而且将会有更多这方面的书)，但几乎没有一本能超越现有的内容，提出有关自己独特模型的术语、表示法或结构。而对于一个初始的对象集合的建立过程来说，这些指导充其量也就是一些比较肤浅的说明。一旦选择了一个错误的对象，就会对整个面向对象开发过程的顺利完成产生巨大的影响。正如最近一个 C++ 讲师对“类”所感慨的：“这些人只是在对象级胡搞一气！他们永远不可能从面向对象中得到任何好处。”

作为本书的作者，我们坚信面向对象的分析和设计是实用的，它能带来实实在在的好处。但我们也相信，用 H. L. Mencken 的话来说，就是“对于每个复杂的、困难的问题，总会存在一个简单的解决方法。而这个说法总是错的。”分析和设计一个庞大的、复杂的软件系统是困难的。面向对象的方法并没有使分析和设计变得比以前的方法更为简单，但它们能帮助我们建立一个更好的产品，能够帮助我们提高生产力。不过它们决不是包治百病的灵丹妙药。

我们认为，那些有实践经验的面向对象分析和设计人员的文献中所缺少的，就是对于一个真正的系统如何进行面向对象的分析和设计。而这正是我们这本书所要讲述的内容。虽然我们给出了事例分析(case study)问题的解决方法，但这个方法本身的重要性远不如这种解决方法的产生过程。我们认为我们主要的贡献在于给出了方法、思路，而不是技术。但很遗憾，那些寻找完美和谐的数学证明的读者必然将会大失所望。

在介绍完事例分析问题之后，我们会给出一个综合 OOA 和 OOD 的方法，用它来开发一个分析模型和设计模型。接下来的章节将分别描述 OOA 模型的每个构造模块(主题层、对象－类层、结构层、属性层和服务层)，以及 OOD 模型的各种部分(问题论域部分、人机交

互部分、任务管理部分和数据管理部分)。其余的章节将对分析、设计和项目管理问题提出一些建议,给出一些说明。

我们使用 Coad 和 Yourdon 技术的表示法建立了面向对象的分析和设计模型,但我们的方法完全不同于 Coad 和 Yourdon 的方法。事实上,我们利用了一些其它面向对象方法中的成份。例如,在讨论对象间的消息时采用了 Ivar Jacobson 的 ObectOry 方法,它能给你以很好的启蒙。我们相信本书中所阐述的规则能普遍适用于面向对象的分析和设计中。

我们还吸取了比较古老的典型的结构化分析方法的优点。在两个事例分析中所应用的实体 - 关系图和事件 - 响应模型证明了它们非常有助于发现相关的对象,并能帮助你更加深入地理解对象之间的相互作用。如果不用结构化分析,我们肯定会迷失得更远。比如说,对于两个事例分析的用户规格说明的纯粹的语言分析就能提供一种非常系统化的方法来标识出候选对象,以用于进一步研究。

当面向对象的学者们热衷于研究他们自己的方法时——如 Booch、Rumbaugh、Jacobson、Shlaer-Mellor,以及 Coad 和 Yourdon —— 我们发现,有可能从有效地解决了某个问题的所有人们那里借鉴一些概念和思想,这种看法也为别人普遍接受。另外,对于比较古老的结构化方法的使用可能会触怒那些自称为是纯粹研究面向对象的人。他们会说,如果你不使用那些在面向对象时代之前所出版的教科书中的指导,也完全可以得到相同的结果。对于大多数软件工程高手来说,他们需要一种更实际的方法,尤其在进度和预算很紧张的情况下需要解决非常困难的问题时。从这个角度来说,盲目地废弃那些能给我们提供很好服务的以前的概念、工具和技术无疑是一种很愚蠢的做法。

由于时间、金钱和精力的限制,我们不可能去研究那些像"星球大战"这种规模的问题,但我们可以选择两个非常现实的事例分析系统,将面向对象分析和设计的规则应用于其中。一个事例分析系统主要是关于响应方面的,这对于工作在实时的、嵌入式系统领域中的读者会有所帮助;另一个事例分析系统主要是关于数据方面的,从事 MIS 系统的读者可能会从中得到一些启发。在本书中我们将使用这两个事例分析系统,它们十分有趣,并可以帮助你了解如何将面向对象技术应用于不同的环境中。当然,在不同的环境中,面向对象方法使用的适用性和适合程度,读者应作出自己的判断。

由于它们的规模要大于那些简单的像"tic-tac-toe(五子棋)"之类的游戏,因此你在一些软件工程教科书中经常可以发现,我们这本书中关于事例分析系统分析和设计的任何严重的问题都会带来很大的麻烦。在过去的几年中,我们这些作者在讨论班和培训班中多次使用了这两个系统,我们一直惊异于我们与学生所讨论的内容以及讨论的深入程度。我们相信,对于这两个事例分析系统的设计已到了可以将其用于实现、编程的地步。虽然我们没有讨论实现的问题,但我们已经对这两个系统建立起了 Visual Basic 原型。然而,限于这两个系统的规模,我们在这儿不提供完整的源代码。

因为这本书并不是用于详细解释面向对象的原则和理论的,所以我们假定读者是对于结构化分析方法和面向对象分析/设计方法有一定了解和掌握的软件工程师。我们假定读者对于 OOA 和 OOD 方面的了解非常有限,至多也只是读过有关这方面的一本畅销书,或者在讨论班或培训班中对它们有过一些浅层的接触。如果具有这样的背景,那么你足可以应付一些"小"问题。但如果你第一次接受一个面向对象应用的工作,那么这些知识可能就显得有些贫瘠了。而这本书就是力图要在 OOA 和 OOD 的初步介绍以及理论在复杂的真

实世界中的应用这两者之间建立起一座沟通的桥梁。

最后，我们希望读者能告诉我们你们是如何运用这些材料、碰到一些什么问题，最后又是如何成功的。我们的 E-mail 如下。我们还给出了 WEB 站点，在那儿我们会随时给出对这些材料所作的修正。

Ed Yourdon
New York City（纽约市）
e-mail：yourdon@acm.org
WWW：http://www.acm.org/～yourdon

Carl Argila
Las Vegas（拉斯维加斯）
e-mail：carl@acm.org
WWW：http://www.acm.org/～aLigra

1995 年 12 月

致　谢

首先我们要衷心感谢 Denver Metropolitan 州立大学的 Joseph Morrell 教授。Morrell 博士总是无私地（并时常放弃睡眠时间）帮助我们完成这部手稿。因为我们要在全世界演示我们的成果，对此，Morrell 博士显示出了无比的耐心，直到负责将这部手稿投入印刷。同时我们还要感谢 Morrell 博士的两位伙伴：系统工程师 Tamara Gillest 和 Beth Ross，他们帮助完成了事例分析系统的 Visual Basic 实现以及用 System Architect 创建事例分析系统的模型。

事例分析系统的 Visual Basic 原始版本是由 Tom McFarren 完成的。这是一项艰巨的工作，对此我们表示万分的感谢。

我们尤其要感谢我们在 Prentice Hall 的编辑 Paul Becker。他每次打电话来总是会问及“……这本书进行得怎么样了……”，这时刻督促我们不要忘记我们还在写一本书。

我们还要感谢 Popkin Software and Systems 公司的 Ron Sherma 和 Steve Schroer，是他们建议我们在这个工程中使用 System Architect 产品。

此处我们可以引用牛顿所说过的一句话，“如果我能比别人看得更远，那是因为我站在了巨人的肩膀上。”而我们正是有了这样一个独一无二的能站在许多巨人——我们那些极具创造力的学生——肩膀上的机会。在过去的两年中，这些学生经历了我们这些材料的不同版本和修正，提出了不少独特的见地。他们非常聪明、勤快，更重要的是，他们能对我们的玩笑付之一笑（当然，是在大多数时间）。附录 T 列出了我们部分讨论班成员的清单。感谢你们，我亲爱的朋友们！

表示法和约定

类 全部以大写字母表示,并以空格分隔。例如:

ARTICLE

COMPLIMENTARY SUBCRIPTION

AUTHOR-ARTICLE TRACK

类的实例(对象) 以大写字母开头,以空格分隔。例如:

Article

Complimentary Subscription

Author-Article Track

对象属性 都用小写字母表示,并以下划线分隔。例如:

article_ title

complimentary_ subscription_ id

arthor-article_ date

类的属性 用大写字母表示,并以下划线分隔。例如:

TOTAL_ NUMBER_ SUBSCRIPTIONS

PAYMENTS_ TO_ DATE

对象的服务 以大写字母开头,以下划线分隔。服务经常使用两个或更多的单词(动词_名词)。例如:

Enter_ Paid_ Subscription

Delete_ Complimentary_ Subscription

Enter_ Article

类的服务 都用大写字母表示,以下划线分隔。类的服务经常使用两个或多个单词(动词_名词)。例如:

CREATE_ NEW_ SUBSCRIPTION

ENTER_ PAYMENT

RECEIVE_ ARTICLE

属性和服务名 必须在它们自己的名字前加上它们所对应的类或对象的名字。例如:

Subscription. Enter_ Paid_ Subscription

SUBSCRIPTION. RECOGNIZE_ SUBSCRIPTION_ REQUEST

Address. address_ details

Subscription Termination. termination_ date

目　录

第1章 引论

没有什么比引进采用一种新事物更难以运用自
如，承担更大风险，成功与否更无确定性的事情了。

——尼科罗·马基雅弗里
《君主论》第六章

1.1 基础：软件开发原理

用“原理”这个词作为一本有关软件开发的事例分析的书的破题似乎有点不同寻常。事实上，有关软件开发的原理不是一成不变的，本书的作者们（他们的软件开发经验已远远超过3000个星期）在长期的职业生涯中，的确是亲眼目睹了其戏剧性的变迁过程。在20世纪50年代，程序是用专门的软件开发方法编制的，每个系统都是独一无二的、专门定制的专用产品。那时根本没有“规范化设计”的概念，即所谓的部件可复用性、可互换性等等。尽管这些系统堪称软件产品的先驱，但它们很难维护及改进。对系统哪怕作一点修改都很困难，而且修改后的系统变得更加难以维护和改进。

到了20世纪60年代，人们越来越意识到以可预测的方式生产可维护的软件是十分重要的，这导致了软件开发原理的第一次大的变革。传统的专门的开发方法逐渐被一种更系统的方法所代替。这种新的软件开发方法，通常被人们称之为瀑布方法，它要求在建立一个软件系统的过程中需要经历一系列规范化的阶段。这些规范化的阶段诸如需求分析、高层设计、详细设计等等，必须按顺序逐一完成，也就是说，在某一阶段未完成之前，其后续阶段不能开始做。而每个阶段的完成则以交付一个或多个里程碑文档为标志。于是，人们提起瀑布方法，总不免想起那一摞一摞厚厚的文档。然而，虽然瀑布方法较之传统的软件开发方法更为规范化，但是，对于那些大的复杂的软件系统而言，这种方法仍然显得力不从心：编制系统的花费总是超过预算，真正所花的时间总是超过预定时间，而且也并不能满足用户的要求。

随着时间的推移，到了20世纪70年代，软件开发原理又发生了一次大的变革。Tom DeMarco在他的名为“结构化分析与系统规格说明”的研讨班教材中，介绍了基于模型的软件工程概念。DeMarco认为，创建复杂的软件系统必须像创建大型、复杂的工程系统那样，首先必须建立系统的书面工作模型，然后，再调集资源去实现系统。按照他的理论，在实实在在的系统还未着手做以前，用户便可以“看到”未来系统的概貌，了解它的功能以及他所想知道的关于系统的一切信息，就好像这个系统已经存在一样。在20世纪70年代，DeMarco的理论是如此的激进，它完全背离了传统的软件开发观念。在那以前，软件工程师们一般是一遍又一遍大刀阔斧地增删代码，当系统显示出几乎能正确执行操作的迹象时便宣称系统成功。如果程序不能正常运行，那也不必担心，只要假以时日，几天，几个月，只要不是几年，

总有一天，系统会成功！

正如图 1.1 所示，基于模型的软件开发方法类似于建筑师们设计大而复杂的建筑物时所采取的策略。建筑师们首先在纸上按比例画出房屋的模型图，这样用户就可以对未来的房屋有比较直观的了解；经过反复的修改，直到用户表示满意后，才开始动手去建造房屋。换言之，这些模型充当了用户、开发者、赞助者和实施人员之间通信和协商的桥梁。

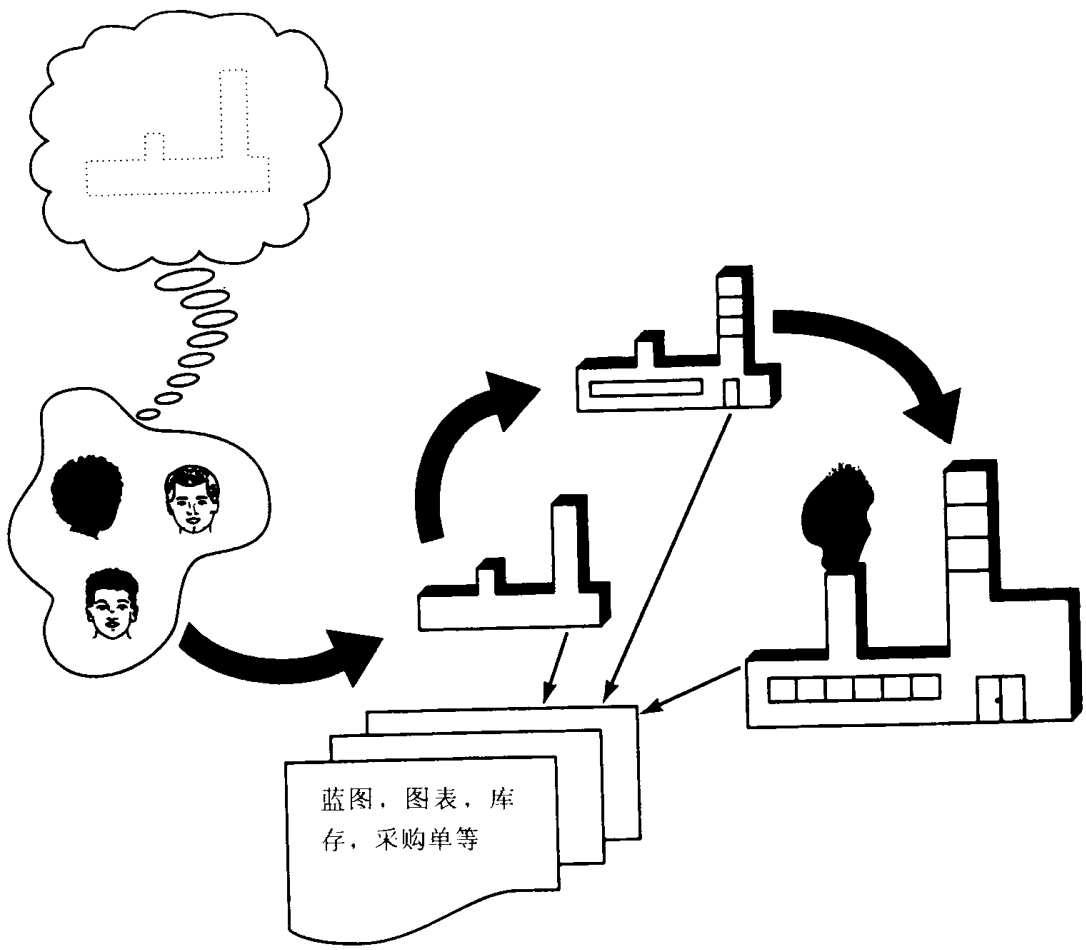

图 1.1　基于模型的软件开发

事实上，现代软件工程都是采用这种基于模型的方法去开发软件的。一个工程和另一个工程之间的区别在于：工程所依据的模型是什么，模型是怎样建立的，以及模型的建立者是谁。在图 1.2 中，我们给出了一个典型的基于模型的软件工程生存期。图中的各种模型是由不同的用户群体为着不同的目的而建立的。例如，需求定义模型是用以获取和处理系统的整体需求的；该模型可以由销售代表以及系统和软件分析员共同建立。

大多数基于模型的软件工程方法都蕴涵着这样一个重要的概念，即事务分离原则(principle of Separation of Concerns)。它主要体现在：分析模型和设计模型是分开建立的。其中，分析模型(这是本书的侧重点)是用于捕捉那些本质的或“逻辑的”系统需求，而不考虑基于实现的或“物理的”系统需求。换言之，分析模型主要是描述系统将要做什么，完全不考虑具体的实现方法和技术细节。而设计模型(这也是本书所要着重讨论的)则是描述在某个