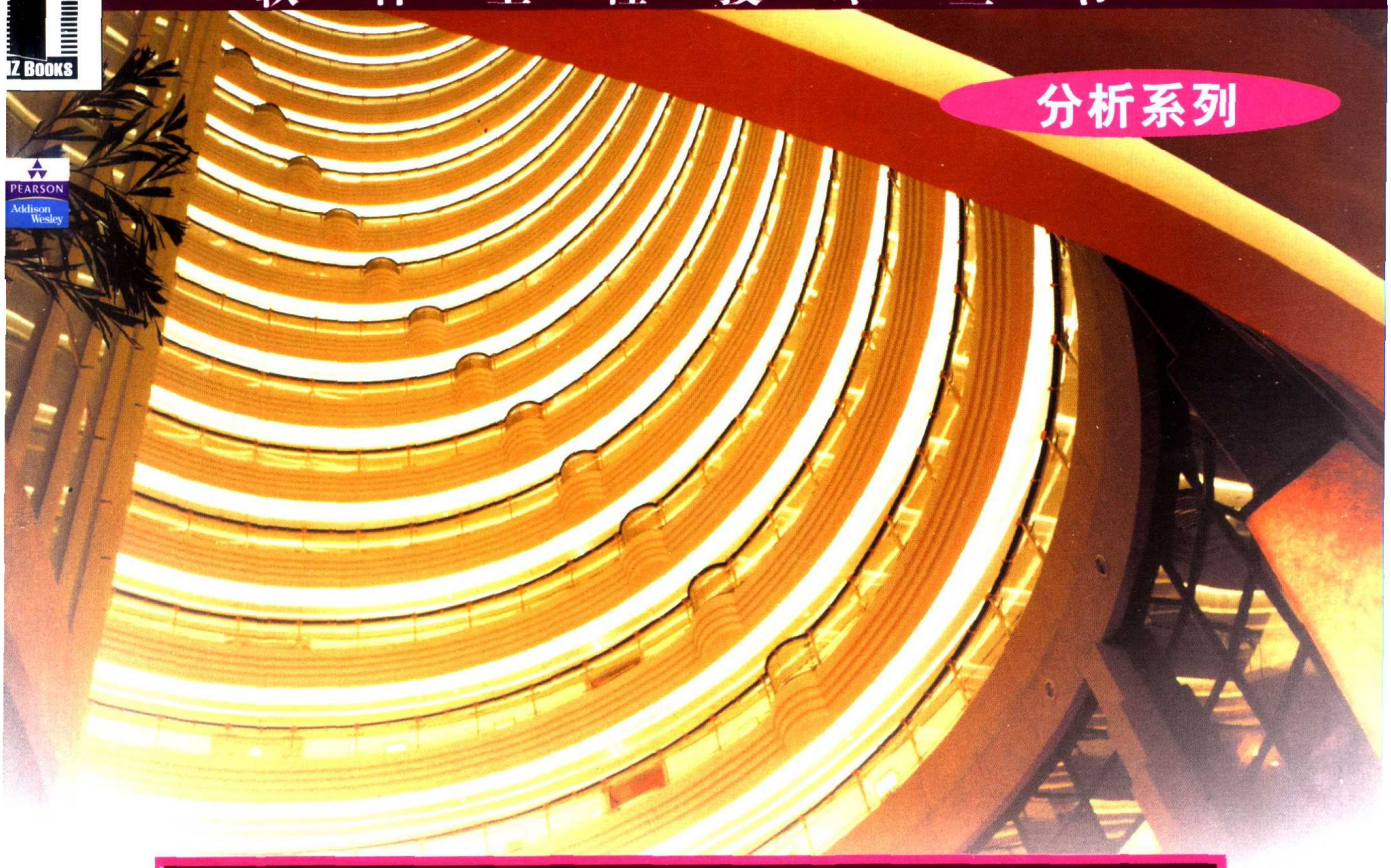
软 件 工 程 技 术 丛 书



分析系列

高级用例建模
卷Ⅰ：软件系统

Advanced Use Case Modeling
Software Systems

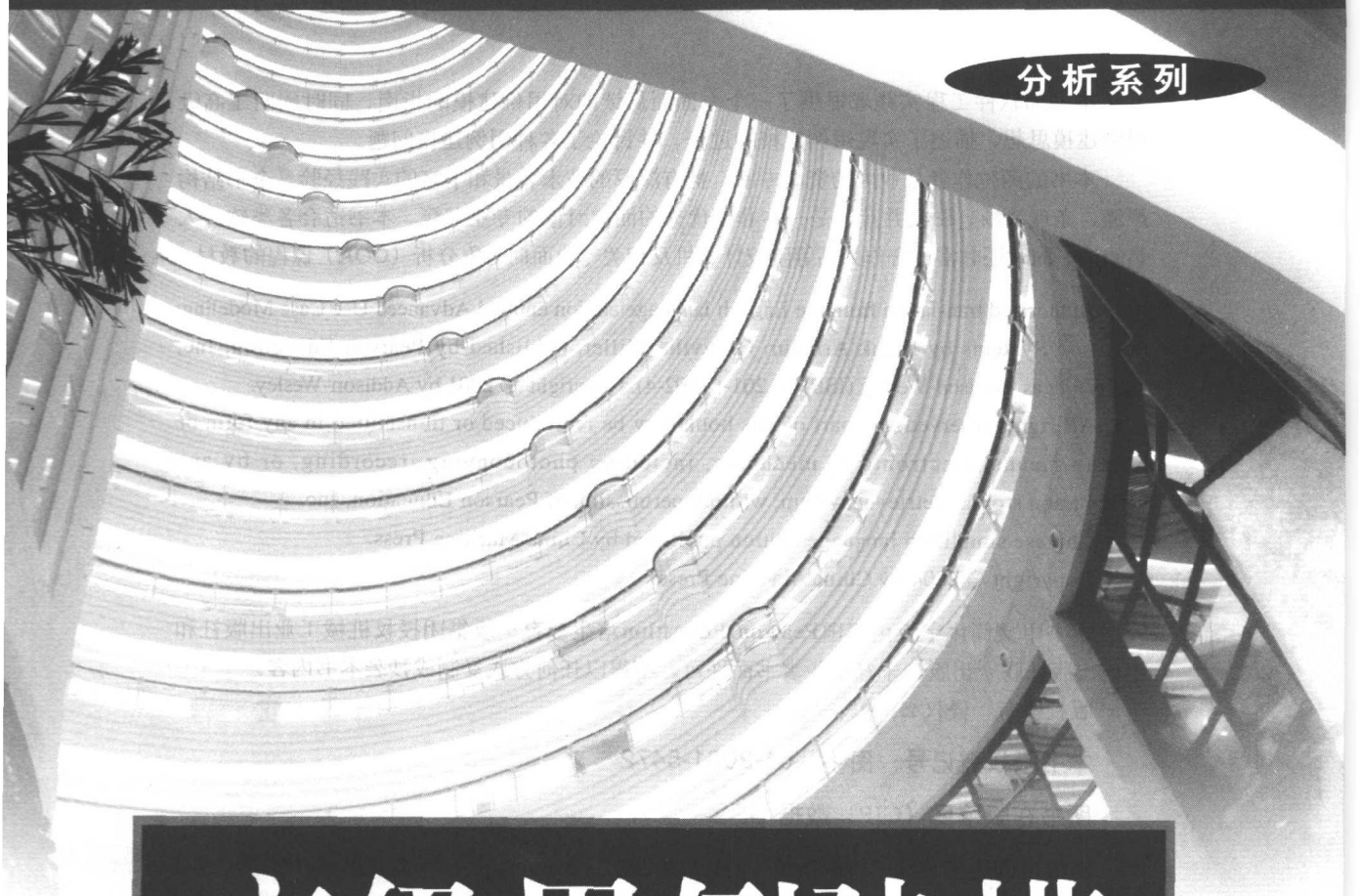
（美） Frank Armour
Granville Miller 著

饶若楠 译

机械工业出版社 China Machine Press
中信出版社 CITIC PUBLISHING HOUSE

软 件 工 程 技 术 丛 书

分析系列



高级用例建模
卷 I：软件系统

Advanced Use Case Modeling
Software Systems

（美） Frank Armour
Granville Miller 著 饶若楠 译

机械工业出版社 China Machine Press
中信出版社 CITIC PUBLISHING HOUSE

本书为软件工程实践者提供了一个全面而易读的对用例建模的指南，同时阐明了高级用例建模思想，描述了实现用例建模的过程，并讨论了各种用例建模问题。

本书的两位作者是业内的资深专家，拥有深厚的技术背景和丰富的实践经验。全书结构严谨、条理清晰、图文并茂，是一本非常优秀的面向对象的专业书籍。本书适合各类软件人员阅读，同时还非常适合作为高等院校计算机及相关专业面向对象分析（OOA）课程的教材。

本书版权登记号：图字：01-2001-5472

图书在版编目（CIP）数据

高级用例建模 卷1：软件系统 /（美）阿穆尔（Armour, F.），（美）米勒（Miller, G.）著；饶若楠译. -北京：机械工业出版社，2004.7

（软件工程技术丛书 分析系列）

书名原文：Advanced Use Case Modeling: Software Systems

ISBN 7-111-14235-7

Ⅰ.高… Ⅱ.①阿… ②米… ③饶… Ⅲ.①软件工程-系统分析 ②软件工程-系统设计 Ⅳ.TP311

中国版本图书馆CIP数据核字（2004）第027908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：王高翔 姚 蕾

北京牛山世兴印刷厂印刷 · 新华书店北京发行所发行

2004年7月第1版第1次印刷

787mm × 1092mm 1/16 · 21印张

印数：0 001-4 000册

定价：40.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

译者序

在应用软件开发过程中，最艰苦的挑战在于捕获需求，即清楚地理解系统必须解决的问题。由Ivar Jacobson于20世纪80年代初提出的用例技术，是由诸多实践证明行之有效的一种用于需求捕获的技术。随着统一建模语言（UML）的应用日益广泛和深入，作为UML建模技术的重要组成部分，用例建模技术在国内也应用得越来越多。用例建模是一种经验性非常强的工程实践技术，入门看起来似乎不是很难，但要准确地把握并在软件开发的工程实践中成功地应用则有相当的难度，需要有丰富的经验。最近以来，国内介绍用例建模的书逐渐多了起来，这是一件好事。但能够以丰富的经验为基础，从实践出发，具备有系统的过程框架和完整的应用实例的书却不多见，本书是其中较好的一本。

本书作者很早就在软件开发中应用了用例建模技术，在本书中他们介绍了很多经验，书中述及的例子和模板都是从实际的软件开发项目中抽取出来的，很实用。本书共分为五大部分。第一部分介绍用例和参与者的基本概念。第二部分针对项目初始阶段讨论如何确定系统边界和平衡用例模型。第三部分阐述高级用例建模的过程框架，讨论启动用例建模工作时应进行的一些准备活动和创建用例模型的初始步骤。第四部分是本书的核心部分，讨论如何对初始用例描述进行更详细的扩展以及如何对所增加的复杂性进行建模，包括开发基本用例描述，详细描述用例的事件流以及相关信息，对扩展、包含、泛化关系建模，将用例映射到对象模型或者其他分析模型，开发实例场景，创建测试用例和文档，组织用例等。第五部分讨论在用例建模中涉及的若干其他主题，包括概念上的用户界面建模，对用例模型中的变更进行管理，对用例建模过程的定制，构造高质量用例模型所必需的一些要素等。本书的四个附录也很有实用参考价值。

译者在翻译本书的过程中力求忠于原著。对于本书中出现的大量专业术语尽量遵循标准的译法，并对关键的概念和有可能引起歧义之处标注了英文原文，以方便读者对照理解。

本书的翻译是在繁重的教学科研工作之余抽空完成的，历时较长。本书翻译的初稿得到了上海交通大学计算机系和上海分布计算技术中心诸多本科学生的大力帮助，在此一并表示衷心的感谢。

由于译者水平有限，书中出现错误与不妥之处在所难免，恳请读者批评指正。

译 者

2004年3月

译者简介

饶若楠，上海交通大学计算机科学与工程系副教授，德国康斯坦茨大学访问学者，Rational公司UML课程授权培训教师。主要研究兴趣包括：分布计算技术、中间件技术、软件工程、网格计算。主持或参与过多项国家及省部级科研项目，是上海市科委信息领域重大专项“面向领域的中间件关键技术及其应用研究”课题总体组专家，在LNCS 等国内外学术期刊和学术会议上发表论文数十篇。

序

当我于1986年提出用例概念时，是以多年的基于构件的系统开发工作为基础的。我们曾用许多其他不同的技术来从事这项工作，这些技术既有重叠也有差距。借助于用例，我们找到了一种有许多特征的工具。这些特征包括：

- 用例是捕获需求的媒介
- 用例是定义功能性需求的基础
- 用例有助于构思应用系统
- 用例对系统的划分定界有所帮助
- 用例是与最终用户和客户交流的手段
- 用例提供了对系统动态的、黑盒的视图
- 用例是对象继承的基础，对象可以自然地通过用例找出
- 用例为需求跟踪提供了工具
- 用例是用户界面和用户体验设计的基础
- 用例有助于将功能性需求转化为对象和构件结构
- 用例是将功能分派给构件和对象的依据
- 用例是定义对象交互和对象接口的机制
- 用例详细说明了对数据库的访问模式
- 用例可以帮助我们衡量处理机的能力
- 用例是集成测试的基础
- 用例定义了测试用例
- 用例是增量开发的基础
- 用例有助于估算项目规模和所需资源
- 用例为用户手册和文档记录提供了基础
- 用例是一种项目控制工具
- 用例驱动开发活动
- 用例已成为表示业务过程的标准方法
- 用例在再工程活动中可用于描述遗留系统的作用
- 用例可以用在对一个业务进行再工程使之成为电子商务的时候。

用例的功用远不止于此。当然，我不会认为用例是软件开发的万金油或银弹。但是，用例思想的开发确实才刚刚起步。

《高级用例建模》一书为软件系统开发用例模型提供了一套指南。它提供了一个技术工具箱，有经验的用例建模人员可以利用其中的技术。正如所有的工具箱一样，每一件工具都有其专门的目的性，不同的工作应当选择合适的工具。**Frank**和**Randy**在提供技术方面做出了杰出的贡献，

充分发挥了他们在这个行业中的丰富经验。

用例持续驱动着业务过程、软件系统和构件工程项目。在这种建模技术后面的一些最初的概念已通过研究人员、参与人员和标准化组织的工作而不断发展，但其基本思想还是一致的。用例是交流业务或软件系统需求的一种很好的方法。我确信，假以时日，新的交流需求将导致其他用例使用方法的发展，前途是无可限量的。

Ivar Jacobson

前　言

我们的客户对用例方法越来越熟悉，有些人甚至只用它们来说明系统。

——Anthony Heritage 和Phil Coley [Heritage 1995]

在目前迅速变化的业务和技术环境下，用例建模已成为定义业务过程和软件系统最主要的技术之一。业务工程师们现在可以利用用例超越业务界限地定义复杂的业务过程甚至整个业务。用例也已成为了一项用于定义软件系统需求的标准，这些系统大多是使用当前的面向对象编程语言，例如Java、Smalltalk、C++等来开发的。在软件构件领域（预计到2001年市场可能达到12亿美金的朝阳产业[Hanscome 1998]），用例正迅速成长为供应商与卖主之间交流的方法。

使用用例技术来定义系统的用户正如这项技术的用途一样多种多样。用例建模早已被多数财富1000强公司使用，并且在全世界众多学术机构中广为传授。这项建模技术正在变得越来越流行。

业务过程和软件需求工程是发展迅猛的领域。这些领域中的研究不断地提出解决潜在问题的新方法，但是实际应用是滞后的，所提出的这些方法往往只有一小部分被采用。这种现象被称为“研究-实践鸿沟”[Berry 1998]，如果想不依靠丰富的经验基础就去写一本关于用例的书只能加剧这种鸿沟。我们的方法是重要的，因为我们展示了一种在现实世界里有稳固基础的实践者的方法。

目标

在过去六年中，我们一直在从事一些软件开发和业务工程方面的艰巨的大项目。为了创建尽可能好的用例模型，我们发现在某些领域有必要对Ivar Jacobson的开创性工作进行扩展。本书详尽地描述了我们所做的扩展，它们补充了Ivar正在进行的工作。用例建模以及描述这些模型的统一建模语言（UML）所具有的灵活性使得我们能提出这些扩展，以便成功地解决现实生活中的问题。

本书的目的是推动用例建模在软件和业务工程中的进步。为了达到这个目标，本书为实践者提供了一个全面而易读的对用例建模的指南。尤其值得一提的是，它同时阐明了高级用例建模思想，描述了实现用例建模的过程，并讨论了各种用例建模问题。

读者对象

本书的读者对象是任何一位涉及软件产品和业务过程的概念化、开发、测试、管理、建模和使用的人士。尽管本书包含大量关于业务过程的内容，但它适用于所有那些身处软件产业的人。用例工程师群体最大的一部分是软件专业人士，因为用例开发最早是作为软件需求媒介被提出来的。

对业务分析师来说，用例工程已发生了巨大的变化。业务分析师和他们从事软件过程的同

行们很快意识到通过软件来实现自动化并不是使用用例的惟一理由。事实上，越来越多使用用例的业务过程建模不再适用于新软件的生产和更新换代，而是用来理解业务，甚至是用来标准化和优化跨越多个业务界线的关键业务过程。

本书所阐述的许多技术可能超越了读者群体所熟悉的软件或业务范围。本书阐述了业务用例和软件系统用例之间存在的联系，并阐明了若干可以从业务过程中导出软件系统的方法。我们惟一要求的是当我们开始讨论软件方面问题时，业务方面的读者能够耐心一点。

学术机构也会发现这本书是非常有用的。本书可以作为面向对象分析（OOA）课程的教材，在这些课程中，用例是关键内容。

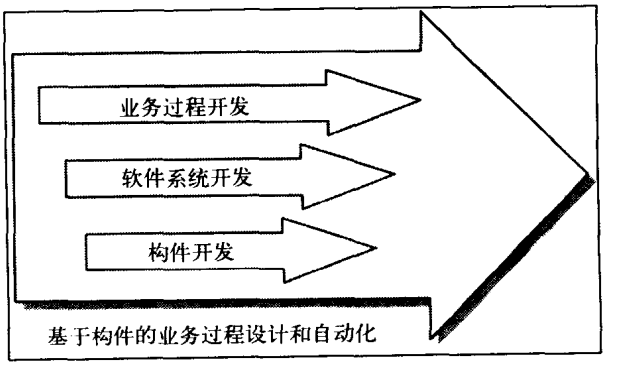
如何使用本书

用例开发的理论往往与用例开发的实践不一致。造成这种不一致的原因之一是，很少有软件开发项目是“崭新的工作”；大多数的项目是从有一个能成功建造软件的遗留过程这样先入为主的观念开始的。我们并不赞成抛弃这些遗留过程，事实上，由于那些需要通过软件开发来解决的问题的特性，这些过程的许多制品可能是必不可少的。另外为了批准启动软件开发项目，有些产品也可能是必需的。

孤立的用例建模难以成功，创建用例模型的过程必须置于具体企业的环境中。每个企业都有独一无二的文化。幸运的是，我们超越了企业的界限，在业务工程和软件开发过程的几乎每一个方面不仅发现了区别，也找到了一些共性。

某个组织中的经验往往可适用于另一个组织，我们总结了一些直接导致用例实践失败的因素。用例建模的缺陷一般可分为两大类：一些是用例开发过程本身的缺陷；一些则是当用例与常用软件开发过程实践相结合时才发现的缺陷。一些缺陷危害极大，它们甚至会导致系统开发终止。

本书提供了一个软件系统建模的过程框架。**过程框架**（process framework）是用于开发一个过程的一整套活动。针对具体的组织，该框架还应该进行定制。本书描述了三个过程框架（见图P-1）中的两个：软件系统的概念化和规格说明。



图P-1 高级用例建模过程的过程框架

每个过程框架都是独立且定义完整的。它们可以协同运作，也可以单独使用。例如，软件系统工程和构件工程可以一起用来为使用构件的软件系统开发提供需求；业务过程与软件系统工程的结合可以创建业务过程自动化所必需的要素。业务过程常常不是通过软件系统实现完全自动化。因此，业务过程的需求是用于执行该业务过程的那些软件系统的需求的超集。

这三个框架提供了一种方法，用来说明业务过程自动化所需的所有的系统工程的需求以及结合软件构造块的需求。当这些过程框架相互结合时，前一个框架产生的输出可能会被用作下

一个框架的输入。

为了充分利用本书，我们建议遵循一个已建立好的软件开发过程。我们理解并不是所有的公司都能以完全相同的方式去遵循一个开发过程。所涉及的**程式**（ceremony），即规范化程度，常常在公司与公司，甚至是项目与项目之间存在巨大的差异[Booch 1996]。

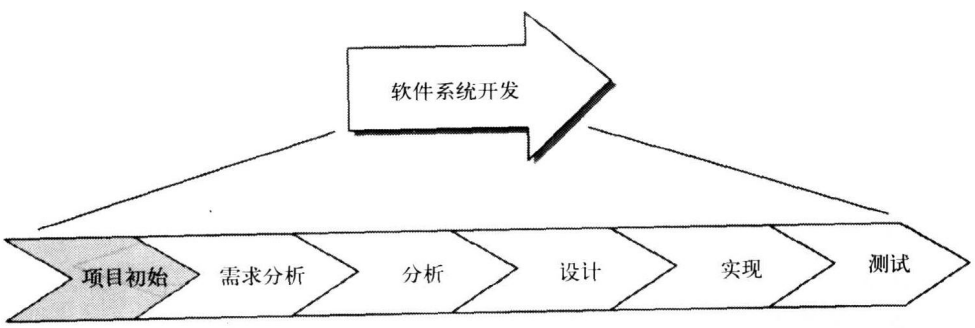
程式有助于定义在多大程度上使用一个过程框架[Miller 2000]。程式高的项目往往会更多地利用过程框架所定义的活动，有可能会大规模地采用高级用例建模。而程式低的项目可能只使用所提到的材料的一部分。但不论程式是高还是低，肯定都能找到某种形式的用例来有效地定义项目的需求。

文章组织与内容

目前市场上有许多关于用例的书。本书的特点在于全面地阐述了用例在软件开发中所起的作用，同时还展示了一些在其他书中找不到的翔实的新资料。我们通过对用例建模领域现有工作的全面总结来权衡这些新资料。

为了使本书能够独立成篇，我们用两章的篇幅阐述用例的基本概念，这两章就是第一部分。第1章论述了用例模型中的参与者的概念性角色，并给出了如何识别参与者的详细说明以帮助用例建模人员发现用例。第2章则讨论了创建用例的通用格式和协议，并说明了统一建模语（UML）——对象管理组织（OMG）关于用例建模的标准。

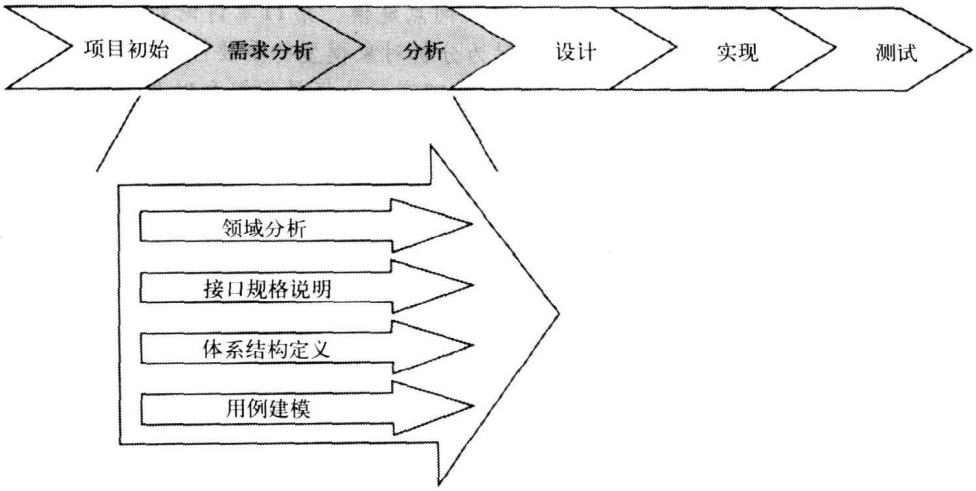
第二部分从软件开发的第一阶段——项目初始开始（见图P-2）。第3章关注此阶段，它着重讨论了那些用于定义系统范围的事物——待解决的问题、由新的或改进的系统所创造的商业机会以及构造一个针对这些机会的软件系统在经济上的可行性。



图P-2 软件开发过程的一般阶段

第4章描述在软件开发过程的需求分析阶段（见图P-3）的用例建模。用例可以描述系统的功能，但是必须要有一种形式来平衡用例模型，而一个设计完备的体系结构提供了这种形式，这一章讨论通过体系结构改进用例模型的一些方法。

在第三部分，我们介绍一个贯穿全书的银行贷款申请的例子来阐明用例建模的思想。这个例子本身并不代表任何现实的贷款系统。为了说明这个例子，现实中贷款申请的必要功能被简化了。



图P-3 需求分析和部分分析阶段的分解

在第三部分，我们还描述高级用例建模过程框架。第5章将用例建模分解为活动组或逻辑相关的活动组（见图P-4），并描述一个用例建模的框架，该框架用于说明一直到第15章的系统用例建模。

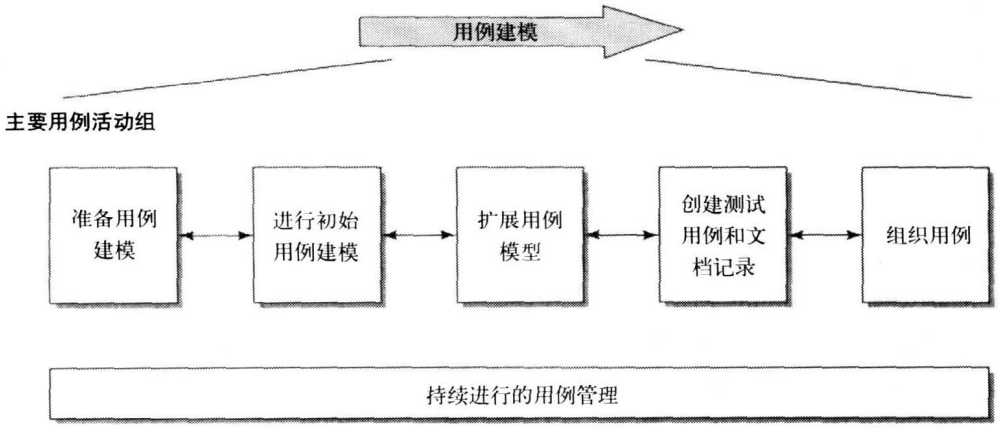


图 P-4 高级用例建模过程框架

第6章描述用例建模准备工作中的初始步骤，概述用例框架的选择和定制、标准和技术的选择以及培训和指导的考虑等。

第7章描述创建用例模型的初始步骤，该活动组的输出是一个用例模型，用以捕获系统需要做什么的“概念上的”全景。

第四部分着重于扩展用例模型。第8章开始讨论如何将初始的用例描述扩展成为有更多需求细节的基本用例，以及如何对这种增强的复杂性建模。第9章讨论将条件和迭代逻辑置于用例的事件流中的实践，并展示对这些概念建模的两种技术。

第10章描述扩展（extend）、包含（include）以及泛化（generalization）关系的用法，这些

关系可用于对用例模型中的选择点、变化点和共同点建模。第11章讨论如何捕获与某个用例有关的附加或补充信息。第12章论述将用例映射为分析对象模型的重要性，并简略说明一些技术，如CRUD矩阵、对象-用例映射表和顺序图等。第13章讨论场景的概念以及如何利用它来补充用例模型。

任何软件工程过程的最后一个阶段都是测试。第14章讨论测试和文档化系统以及用例在驱动这些活动方面所起的作用。第15章分析如何使用业务功能包以及如何使用基于用例的前置条件和后置条件的依赖关系来组织用例，并简要总结关键用例制品。

从第16章起是第五部分——其他主题。第16章分析用例对用户界面设计的影响。事务可用于划分用例模型，从而为概念上的用户界面开发提供要素，分组技术则使根据事务来构造屏幕画面成为可能。

第17章分析变更对用例模型的影响。在成功的软件系统中，影响系统功能的变更是难以避免的。变更可能发生于项目开发期间或项目完成后。

第18章讨论实施高级用例建模所必需考虑的一些事项。应该根据项目的需要来决定使用过程框架的全部还是一部分，这一章概述决定使用程度的要素，并描述如何文档化该过程。

最后一章，第19章，讨论一个好的用例模型的质量特征，并描述用例建模在系统分析工作中可以扮演的不同角色。最后，简述用高级用例建模进行迭代和增量的开发。

补充材料

本书独立成篇，阅读本书可以不必参考其他著作。但是，在需求工程、用例开发和过程改进等方面，还是有许多有用的资料。

软件开发过程

- Ivar Jacobson, Grady Booch, and James Rumbaugh, *The Unified Software Development Process*, Addison-Wesley, Reading, MA, 1999.
- Dean Leffingwell and Don Widrig, *Managing Software Requirements: A Unified Approach*, Addison-Wesley, Reading, MA, 2000.
- Geri Schneider and Jason P. Winters, *Applying Use Cases: A Practical Guide*, Addison-Wesley, Reading, MA, 1998.
- Rational Software Corporation, *Rational Unified Process*. 2000.

业务过程工程

- Ivar Jacobson, Maria Ericsson, and Agneta Jacobson, *The Object Advantage:Business Process Reengineering with Object Technology*, Addison-Wesley, Reading, MA, 1995.
- Michael Hammer and James Champy, *Reengineering the Corporation*, Harper Business, New York, 1993.
- Rational Software Corporation, *Rational Unified Process*. 2000.

构件开发

- Ivar Jacobson, Martin Griss, and Patrik Jonsson, *Software Reuse: Architecture, Process, and Organization for Business Success*, Addison-Wesley, Reading, MA, 1997.

• Clemens Szyperski, *Component Software: Beyond Object-Oriented Programming*, Addison-Wesley, Reading, MA, 1998.

你可能注意到在本书中有不少参考资料。在本书之前出版过一些有关用例方面的著作，我们对此做过广泛的调查，并发现了许多值得借鉴的观点。我们发现在许多方面，我们独立得出的结论与那些著作的结论是相似的。对这些情况，我们给出了最早提出这些观点的著作的参考书目。这不仅方便读者通过这些参考书目获得其他信息，而且也让读者知道这些贡献应该归功于那些当之无愧的作者们。

如果你想获得关于用例的最新信息、辅助或补充材料或者与作者的联系方式，请访问我们的网站：www.advancedusecases.com。

致谢

本书主要以Ivar Jacobson的工作为基础，他是用例工程领域的开创者和预言家，在本书中我们努力忠实于他的原著。我们感谢Zina Znayenko Miller对本书不知疲倦的校订，他往往连一个小注释都不遗漏。我们也要感谢我们的审阅者们，他们，尤其是Monica Gupta和Todd Hansen，在本书的各个阶段提供了有价值的反馈。如果没有他们，这本书是不可能达到目前的程度的。

我们感谢美国管理系统公司（AMS）及其客户，他们为我们提供了许多用例建模的机会与经验。我们也感谢以下这些人：Andy Baer、Chris Ball、Jeff Bitner、Susan Bowler、Lorrie Boyd、Mike Bradley、Bob Brodd、Bill Catherwood、Judy Cohen、Dennis de Champeaux、Peter Dimitrious、Sean Furey、Mary Gorman、Kevin Heineman、Peter Knowles、Steve Larue、John McGregor、Les Moore、Perri-Ann Sims、Mark Schroeder、Chris Tatum以及Patrick Wall。我们也要感谢Brenda Damario、Christine Milliken以及美国人口普查局的Nora Parker和Enea Business Software AB公司的Karin Palmkvist。

我们还要感谢Addison-Wesley出版公司的评阅者们,他们是：Jeff Bitner，AMS公司合作技术组的资深负责人；Maria Ericsson；Monica S. Gupta，AMS公司先进技术中心的中间件与Web集成实验室的前任主管；Todd Hansen，Make Systems公司的资深架构师；Mark Schroeder，Folio[*fn*]公司；以及Sam Supakkul，Digital Pockets公司的资深架构师。

我们还要对美利坚大学和乔治·梅森大学所有参加系统与需求分析课程的学生表示感谢，他们的经验与见解帮助我们完善这本书。这项工作的早期版本曾在OOPSLA'98的讲座上展示过，该讲座的听众提出了一些十分发人深省并影响本书的问题。我们还要对美利坚大学的Sarah Alijani和美国参议院警卫部队的Steve Kaisler的贡献表示感谢。

我们同样感谢那些与本书出版有关的人，Kristin Erickson、Krysia Bebick、Carter Shanklin以及Addison-Wesley出版公司的所有工作人员。特别感谢项目经理Diane Freed和为本书排版的Kim Arney。最后，我们感谢这个行业中愿意与他人分享自己经验的人。

我们希望这本书会像对我们一样，对您和您的企业有所裨益。

Frank Armour
Granville Miller
2000年11月

绪　论

知识生产力将成为确立一个公司、一个产业、一个国家的竞争地位的决定性因素。

——Peter F. Drucker [Drucker 1993]

业务的本质在变化，这种变化就发生在我们的身边。起源于工业革命并且在近三百年来不断完善的业务基本方法已被永久地取代了。新的方法主要基于信息与技术，并为全球竞争所驱动。在这个称为信息时代的纪元中，生产的方式是基于知识而不是基于劳动力和竞争。信息时代的先锋们创建了许多新公司，它们是如此强大，其实力足以匹敌并威胁原产业界的巨头们。

在数量上，这些新业务方法比呈现在我们眼前的要多得多。信息与技术是至关重要的，但并不足以在业务范围内孕育如此大的变化。一个小小的刚起步的公司，如Amazon.com，仅依靠技术是难以挑战Barnes and Noble，但它们现在能够利用技术与信息来增强生产力和提高竞争优势，并通过一种有组织、有创意的方法取得了竞争上的优势地位。

生产力的增长和竞争的优势对我们中间那些尚未成为信息时代先锋的人也具有影响。计算机与电信技术的高速发展对商业的发展速度有着广泛而显著的推动作用[Greenspan 1999]。我们大多数人似乎都从中受益了，即使我们并没有亲自使用那些促使10%的先锋者走向成功的特别的方法体系。但是，理解促成我们目前正在经历的这种普遍繁荣的关键因素是很重要的，我们还必须了解那些能使我们更具竞争力的因素。

信息技术的作用

信息技术在当今公司里所起的作用是毋庸置疑的，它是每一个公司运作的基础。关于业务与计算机的关系，虽然人们时有讨论，但已有定论，即它们之间是一种能促进或引起业务变更的合作关系。

信息技术关注于业务的变更，业务本身也是如此。业务工程和自动化不再等同于大量裁员[Hammer 1995]。在许多情况下，自动化可简单地被用于处理普通的工作，使我们人类能专注于我们工作中更有趣的领域。有趣的是，也有自动化不适用的例子，在这些例子中，因自动化而引发的问题比它能解决的还要多。许多医疗保险系统已经全部自动化了，只有在对付欺诈行为时才需要人的干预。因此，在现今的医疗保险系统中，人与技术是和谐协作的。但是，我们应当意识到变更是不可避免的。如果在竞争优势能够加强时我们不思改变，而我们的竞争对手在求新进取，那么我们将会丧失我们一直为之苦苦拼搏的市场地位。

信息技术是这种业务变更的基本促进者[Hammer 1993]。然而，技术创新不能凭空进行，它们是在业务系统的不断变更过程中完成的。这种变更起源于组织、过程和人员的创新精神，技术可以被用于促进这种变更。

通过信息技术促进变更

我们找不到比万维网更能展现信息与技术间协作关系的例子了。很少有希望在行业中保留

一席之地的公司能承受得起因忽视这种媒介而导致的损失。作为销售、市场和客户支持的一种途径，因特网依旧利润丰厚。事实上，拥有网站的公司的数目已超越了拥有免费咨询热线的公司的数目！网站只不过是软件自动化增进效率的众多表现方式之一。

有一种流行的观点认为：所有使用信息技术的业务变更项目都应有显著的效果。当然，这种显著的效果对任何企业都十分重要，而且被普遍视为投资上的回报。但是，如果我们仅仅把这条底线作为所有项目的指标，就会重蹈20世纪的许多重大错误的覆辙。创建基于Web的技术中心的初衷并不是为公司谋取可见的利润。你永远无法预见一个重要的技术变更会发生在何时，对变更做好准备的企业比那些没有做好准备的企业将更可能成功。

信息技术另一个切实的好处就是创新的潜力。创新对信息时代而言至关重要，但是，有关如何以一种系统化的方法来促进创新的指导却很少。Peter Drucker，这位智者曾在1993年从经济学与社会学的角度成功地预测了许多我们今天所见的东西[Drucker 1993]，他描述了现在被我们称为知识管理的动力。他声称提高知识生产力的途径在于：

> "……将注意力集中在最终结果上，集中在任务上，集中在工作上。'Only connect'是伟大的小说家E.M.Forster的不断告诫，它不仅是艺术家的特质，对于达尔文、波尔、爱因斯坦等科学巨匠而言也是如此。在他们那个层次，结合的能力也许是天生的，是被我们称为"天才"的谜的一部分。但从更大范围来说，结合并从而提高现有知识产出的能力（无论是对于个人，团队或是整个组织）都是可学习的，最终，它也应是可传授的。这需要一种定义问题的方法体系，这种需求也许比对当前流行的"解决问题"的方法体系需求更迫切。"

Peter Drucker所描述的是一种系统地定义在业务中发生的问题的方法体系。没有这种方法体系，我们就达不到创新所需要的融会贯通。我们需要一种方法体系，它能使持续利用技术和信息来增强生产力与竞争优势的能力得到深化。这些生产力增长的价值，一些可能是有形的，而另一些也许是无形的（至少从严格的会计规则角度来说）。

我们定义问题的能力赋予我们三块非常重要的生产力拼图的拼板。第一块也是最重要的一块是，专注于我们期望取得的结果的能力。其次，是对不断学习的渴望和对需求的认识。最后，是解决有价值的问题之后的满足感。在全球经济和科技进步的激烈竞争中，一个有动力的、不断学习的企业将处于有利地位。

软件产业

软件产业已被证明是一个独立的、利润丰厚的产业，它为某些人带来了无以伦比的财富。这些财富与机遇也许来自于技术在业务变更中所起的作用。当然，并非所有的软件都是瞄准业务，一些软件瞄准了客户而另一些只是出于兴趣而编写的。

不管客户是谁，构造满足他们需求的软件是系统成功的关键。因此，如同在业务中一样，在软件中明确而规范地描述问题是非常重要的，同时也是非常困难的。Fred Brooks[1987]在一篇著名的文章中写道：

> "构造软件系统最困难的部分在于精确地决定要构造的内容。*没有比建立详细的技术需求（包括面向人、机器和其他软件系统的所有接口）更困难的概念性工作。*如果这部分工作出了差错将会严重削弱所生产的系统，而且在后续工作中很难加以调整。"

Brooks那时提出的观点直到今天仍然正确。对一个成功的软件解决方案交付产品来说，最困难的部分是：待构建系统的精确的概念和规格说明。这不仅需要深入地理解所要解决的问题，还要理解如何用计算机来帮助解决问题。

软件应用系统的概念和规格说明所带来的问题在于不可见性[Brooks 1987]。不可见性意味着使新的软件功能可视化是非常困难的。客户试图描述他所想要的，它们作为需求被收集起来并提交给热心的开发者。不幸的是，在这个过程中，有些信息常常会在“翻译”时被遗失了（见图I-1）。这种不完善的交流也是为什么说明一个应用系统如此困难、要费那么多时间的原因之一。如果不是客户所要求的，新的应用程序往往是无用的。对需求进行理解的复杂程度自然随着系统的规模、问题和解决方案所需的创新性及系统接口的数目而增加。

图I-1 确切的软件功能难以可视化和交流

软件构件产业

由于软件产业的竞争性，进入市场的时间常常是系统成功与否的关键因素。构件是一种可以提高软件生产效率的充满前途的技术。构件使得用基本单元组装软件系统成为可能，系统开发人员使用这些基本单元建立基本的系统，然后在此之上添加有竞争优势的自己的东西。

确实，捕获基本单元的技术显然是有效的，JavaBean是一个广泛使用的构件技术的例子。创建构件最关键的因素在于：如何在对构件创建者有用的细节层次上精化需求[Finch 1998]。在该层次上精化需求应着重于解决问题而不是过度工程。

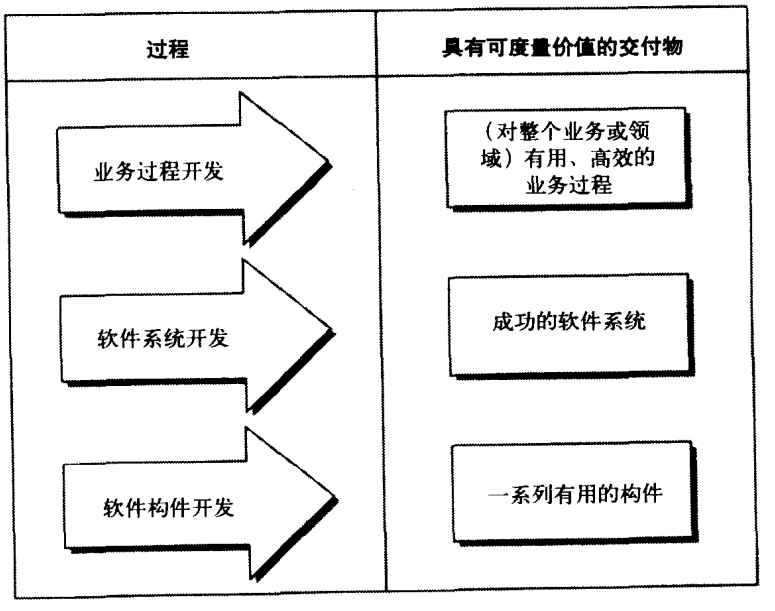
可度量的价值

采用高级用例建模的工程实践主要针对复杂系统的开发。这个复杂的系统称为**最终交付物**（final deliverable）（见图I-2）。最终交付物的价值应该是可度量的，这样才能理解工程的投资回报。在通往最终目标的道路上，应该建立模型以便更好地理解目标，这些模型称为**制品**（artifact）。

当我们把具有可度量价值的系统交付给某些人时，这些人称为项目相关人员。**项目相关人员**（stakeholder）是指那些受复杂系统的交付物直接影响的人。有的项目相关人员直接和系统交

互，他们可能是用户、操作员、安装人员等等。这些人在用例模型中称为**参与者**（actor）。参与者是一个和系统交互的实体（有时可能是人）。我们将在第1章中详细阐述软件系统的参与者。

有些人可能不被系统的交付物直接影响，这些项目相关人员可能是那些不使用该系统的客户、管理者、开发人员等等。无论项目相关人员是直接的还是间接的，这组人员是否满意对于项目的成败是至关重要的。然而，每一组人员对系统有不同的期望，也有不同的观点。



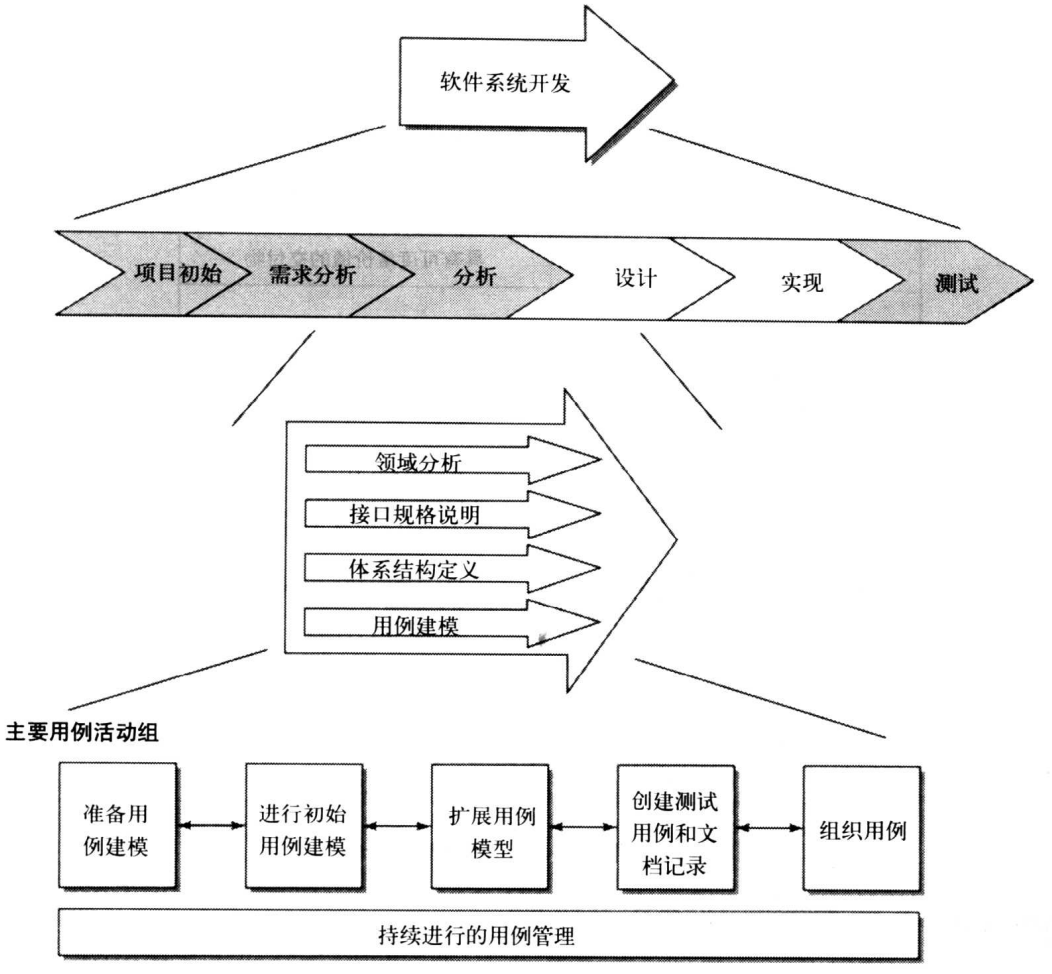
图I-2 高级用例建模过程的最终交付物

高级用例建模

高级用例建模是一种规范地说明业务工程、软件工程（见图I-3）或构件工程的系统化的过程。这个过程是相当强大的，而且易于被关注业务或软件的人理解。这种技术以系统的使用为目标，注意力直接集中在将要解决的问题上。这种方法体系阐述了如何创建模型，这些模型说明了在业务、软件系统或构件系统中的交互。对交互建模使得我们无论在什么领域，都可以获得有形的可测量的价值（例如，创新能力和处理意外事件的能力）。

在高级用例建模过程中，用例是建模的基本单元。用例驱动过程中的其他活动及其后续活动。最重要的是，用例提供了一种通用的语言，组织的各部分（业务工程、软件系统工程和构件工程）通过这种语言可以相互交流。

用例提供了一种增量式和“模块化”的方法来描述一个系统。**用例**（use case）描述了一个复杂系统的用户使用该系统的方式。对模型来说，系统功能的添加或变更都是容易实现的。一个用例仅仅描述了使用系统的一种方式，对系统完整的描述需要许多用例。在一个给定系统的范围内，用例和参与者的结合形成了该系统的用例模型。**用例模型**（use case model）描述了一个系统的组合行为。



图I-3 高级用例建模过程框架

用例表示了系统需求，不同的项目相关人员都可以理解这种表示。每个用例用叙述式的文本来描述，用例模型则用叙述式文本和一组易于掌握的符号（称为UML，统一建模语言）的子集相结合来描述。

高级用例建模是一种可以解决出现在业务工程、软件工程和构件工程中的大量问题的方法体系，这一方法体系的成功使用需要一个环境，在这个环境中最终结果会体现出价值。

小结

高级用例建模是在20世纪80年代末涌现的众多技术（业务过程、面向对象系统、软件构件工程等）的基础上发展起来的，该方法体系是用于定义、概念化和说明上述领域中的问题的一种系统技术。明确且规范地描述这些领域中的问题及其解决方案有许多明显的好处——更有效的业务过程，新的软件系统或构件库。