Addison-Wesley
对象技术丛书

PEARSON
Addison
Wesley

用例建模

USE CASE MODELING

[美] Kurt Bittner
Ian Spence 著

姜昊 许青松 译

Foreword by **Ivar Jacobson**

清华大学出版社

Addison-Wesley 对象技术丛书

用 例 建 模

[美] Kurt Bittner
Ian Spence 著

姜 昊 许青松 译

清华大学出版社
北 京

内 容 简 介

本书提供了识别和描述用例的实践细节和用例的详细说明，是对 Ivar Jacobson 著作的扩展和补充，完整地介绍了确定用例以及用例发展情况的细节。

本书可作为软件学院及大学计算机等专业相关课程的教材，也可以作为软件开发人员参考。

EISBN: 0-201-70913-9

北京市版权局著作权合同登记号　图字：01-2003-5206

图书在版编目（CIP）数据

用例建模/[美]比特纳，[美]斯彭斯著；姜昊，许青松译.—北京：清华大学出版社，2003
(Addison-Wesley 对象技术丛书)
书名原文：Use Case Modeling
ISBN 7-302-06850-X
Ⅰ.用…Ⅱ.①比…　②斯…③姜…④许…Ⅲ.①软件工程－系统分析　Ⅳ.TP311

中国版本图书馆 CIP 数据核字（2003）第 05933 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

文稿编辑：姜汉鲁
印 刷 者：清华大学印刷厂
发 行 者：新华书店总店北京发行所
开　　本：170×230　**印张**：17.75　**字数**：385 千字
版　　次：2003 年 8 月第 1 版　2003 年 8 月第 1 次印刷
书　　号：ISBN 7-302-06850-X/TP・5081
印　　数：1～4000
定　　价：36.00 元

序

自从我在 1986 年首次提出用例的概念以来，用例的使用已经经历了较长时间的发展。面向对象的编程技术清晰地体现了用例的价值和力量。用例既对面向对象范例的发展做出了贡献，同时也从中得到发展。目前，对于一个人理解和应用 UML 以及其他现代软件过程、如 RUP（Rational Unified Process），用例方面的知识很重要。

当有效地使用用例时，用例作为软件过程中需求活动的重要组成部分，具有显著的价值。这些用例极大地增强了开发团队与涉众之间的沟通能力，并且使我们能够更容易更准确地确定需求。

在帮助开发团队了解系统应向其涉众所提供的价值方面，用例的作用是独一无二的。因为用例可以描述用户使用系统的方式，以及系统可以为那些用户所提供的服务，用例提供了一种独特的方式来在关于系统必须做什么的问题上达成一致。达成一致对于项目的成功是至关重要的：如果涉众不能在系统必须取得何种价值上达成共识，那么项目就不可能获得成功。

因为用例可以帮助人们达到这种理解，故很自然地就提供了一种划分项目活动的结构时要遵循的原则。对于项目组中的许多人员来说，用例都扮演了重要角色——使用系统需求的分析人员；应用用例到设计和开发系统的过程中的开发人员；验证系统是否传递了涉众所需价值的测试人员；记录如何使用系统的技术作家；以及帮助简化系统使用方法的用户经验专家。所有这些项目的团队成员都必须理解用例，以便可以开发出更好的解决方案。

目前，在用例建模的文献中缺少了一些内容：对识别和描述用例在实践中日常细节的描述。本书提供了这些细节，给出了用例模型的定义，并提供了用例的详细说明。本书是对我的早期著作的扩展和补充，完整地介绍了如何确定用例以及用例是如何演化的。

《用例建模》一书以基本概念为基础，着重介绍了 Kurt 和 Ian 在各个行业中多年工作的实践经验——他们在开发团队中工作，既做过顾问，也做过团队成员。他们将积累的经验应用到了这种具有实践性和需要洞察力的工作中。对于初学者来说，本书是极好的入门书籍。对于使用用例富有经验的编程人员来说，本书可以作为日常使用的参考。

本书是迄今为止，在用例领域最棒的书。阅读这本书就可以理解用例的思想，并根据构建系统的种类以及团队成员的成熟度，将这些思想与常识结合在一起应用。

Ivar Jacobson

2002 年 7 月

前　言

何为用例

在这个世界上，似乎我们有太多的事情要去做，有太多的事情要去思考，那么需要做的最后一件事就是必须学习新事物。据 Eric Sevareid 的观察，问题的主要起因是解决方案。

而用例恰恰可以解决带有需求的问题：如果具有严格的声明需求，则很难描述事件的步骤和序列。为了找到原因，我们来分析一个简单的示例：

示例

自动柜员机系统必须满足某些需求：

1. *系统应该允许客户从他们的账户中提取现金。*
2. *系统应该确保客户的账户不会透支。*
3. *如果客户试图透支，那么系统将只允许客户透支指定的数额，并收取交易费用。*
4. *如果客户正在使用一台自动柜员机（ATM），而这台柜员机又不是该客户账户所属金融机构的，那么就要在账户中收取一份额外的费用。*

够简单的吧，是不是？

这些步骤应该以何种顺序进行呢？顺序不同会有关系吗？如果 ATM 并不属于客户账户所属的金融机构，那么 ATM 的使用费用应该在检查透支之前收取还是在其之后收取？如果客户的账户余额少于 ATM 的使用费用，那么在核对透支之前收取 ATM 使用费，就会自动导致提交透支收费操作，即使客户决定取消交易，情况也是如此。这种做法对吗？如果仅有声明需求——这是许多项目仅有的信息，那么就很难说这种做法是否正确。

简单地说，用例可以将事件序列的说明放在一起，引导系统完成有用处的任务。和听起来一样简单——这也是很重要的。在面对很多需求的时候，通常不太可能理解需求的作者真正希望系统完成何种任务。在前一个例子中，通过指定特定行为发生的时间和条件，用例减少了需求的不确定性。这样的话，行为的顺序就可以当作是一种需求。用例特别适用于捕捉这类需求。尽管听起来很简单，事实情况却是由于常规的需求捕捉方法所侧重的是声明需求和“应该怎么怎么样”的陈述，所以完全无法捕捉系统行为的动态方面。用例是一种简单而有效的表达系统行为的方式，使用这种方式所有涉众都很容易理解。

但是与任何事物一样，用例也存在自己的问题——在用例非常有用的同时，人们也可能误用它，结果就产生了比原来更为糟糕的问题。本书的主题就在于此——*如何有效地使*

用用例，而又不会产生出比原来更严重的问题。

谁应该关注用例

对这个问题最简短的答案就是“任何人”，或者至少是在交付一个可以满足客户需要的系统中的某些方面所涉及到的任何人。为了对谁应该关注用例的问题有一个更明确的认识，以下列出了可以从描述系统行为的用例技术中受益的人：

- **客户**——要确保正在构建的系统正是他们想要的系统。
- **管理者**——需要对系统的功能有较全面的理解， 以便对项目进行有效的计划和监督。
- **分析人员**——需要对系统的功能进行描述和记录。
- **开发人员**——需要理解系统需要的功能，以便进行开发。
- **测试人员**——需要知道系统将要完成何种功能，以便对系统进行检验。
- **技术作家**——需要知道系统将要完成何种功能，以便可以对系统进行描述。
- **用户经验设计人员**——需要我们理解用户的目标，以及用户使用系统来达到目标的方式。
- 还有在实际构建系统之前，希望更好地理解需要构建哪些功能的所有人。

如何阅读本书

本书主要讲述了如何创建用例模型的问题，而更重要的是关于编写用例详细描述的问题。为了保持对这个任务的关注，我们有意地省略掉使用用例的项目生命周期部分，而没有直接写入这些内容。这些领域包括用户界面设计、分析、设计、技术写作、测试和项目管理。其他作者对这些领域提供了充分的阐述，我们觉得如果将精力仅仅集中在用例本身的问题上，那么作为读者可以获取最大的收益。希望能够认同这一点。

本书旨在成为那些真正从事这项工作，并发现使用用例过程中特殊问题的实践者手中便捷的参考书。大家可以通篇阅读，但本书的真正意图是为读者提供一些在第一次阅读之后，仍能够继续使人受益的内容，并成为读者可以随手可得的“导师”。书中列出的主题来自于无数项目团队的工作经验，而这些项目团队曾遇到与你相同的问题。

本书分为两部分。在第 I 部分“用例建模入门”中，介绍了一些用例建模中的基本概念，为了有效地使用用例，大家需要理解这些概念。在第 I 部分结束时，给出了开始使用用例的一种极好的描述：工作室。

- 第 1 章“用例建模简介”，为那些不熟悉用例，或那些在阅读了其他书籍和文章之后，仍然发现自己对基本概念产生混淆的人提供了实践背景。目的是给出对用例方法的简要概述，而不去涉及很多正式的细节。
- 第 2 章“用例建模基础”中给出了一些用例建模技术所依据的基础。其中所介绍的概念可以作为书中后续章节内容的基础。
- 第 3 章“建立构想”提供了一些重要工具，可以用于确定需要解决的业务问题、确定解决方案中的涉众，并可以确定对于涉众将要解决的业务问题，系统应该完成的任务。如果我们在开发自己的用例模型时要定义正确的解决方案，那么这些信息将会非常重要。
- 第 4 章“确定参与者和用例”介绍了识别用例模型中关键元素的过程和细节。介绍这些内容的目的是提供对参与者和用例基本概念的正确理解，帮助完成那些有时会令人迷惑的任务。
- 第 5 章“用例建模工作室”介绍了开始使用用例的实用性，包括如何运行用例工作室和如何处理开始使用用例的实际细节。

在第 II 部分“编写和评审用例描述”中，研究了使用用例更为详细的细节，其中包括对用例的分析、编写用例描述的方式（并不是第 I 部分中所给出的简单但并不完整的描述），以及在实践当中使用用例的意义。在这些章节中，对编写详细用例描述进行了深入的研究。

- 第 6 章“用例的生命周期”中介绍了用例从概念发展到完整描述过程中所经历的变化。本章为后面的章节建立了一种环境，并将第 I 部分中的内容展开到一个更大的环境中。
- 第 7 章“用例的结构和内容”介绍了用例中各种不同的组成部分——基本流、前置条件、后置条件和备选流，以及一些相关主题。
- 第 8 章“编写用例描述：概述”介绍了编写详细用例描述的目标和挑战，并提供了一些成功控制这种具有挑战性任务的策略。
- 第 9 章“编写用例描述：修订”中介绍了如何编写用例描述、如何处理细节以及为了可读性而如何划分描述结构的一种机制。使用了一个经过演化的示例对这些机制进行讨论，在示例中逐步地、系统化地应用了各种技术，用于提高用例描述的品质。
- 第 10 章“存在的问题”介绍了一些大多数团队在使用用例间关系（尤其是包含关系、扩展关系和泛化关系）和参与者间关系时，所遇到的一些问题。
- 第 11 章“评审用例”中介绍了组织和管理用例模型的评审，其中包括对需要特别注意领域的总结。
- 第 12 章“总结”中涉及到了一些关于如何在较大项目环境中使用用例的主题，

作为对用例使用问题的总结。以这种方式，为读者提供了许多信息来源的参考，可以从中找到关于在其他规则中使用用例的更多信息。

致　谢

多年来，我们一直很荣幸地与许多同事和客户共事，在形成这里所介绍的观点的过程中，他们提供了很多帮助。在这里不可能将这些人一一列举，但我们尤其要感谢那些在用例方面贡献自己观点的同事。我们要感谢 Ivar Jacobson，他提出了用例建模的概念，并最在现代软件开发过程中早定义了用例的作用，非常感谢他对这个项目中的支持和鼓励。我们还要感谢 Dean Leffingwell，他定义了用例的作用和传统需求管理方法。我们还要感谢 Bryon Baker，Chris Littlejohns，Anthony Kesterton，Gary Evans，Laurent Mondamert，Peter Eeles，Brian Kerr 和 Susan August，感谢他们在本书长期改进的过程中，对各个阶段所提出的赋有洞察力的建议。我们要特别感谢 Douglas Bush 和 Ida Audeh，感谢他们的协助，使我们可以在编写本书的过程中做到清晰简洁。我们还要感谢 Rational 公司的技术顾问，为编写本书，他们提供了自己的经验并提出很多问题。最后，我们要感谢那些与我们以及这些顾问共同工作过的客户，因为他们的经验和问题使我们意识到编写本书的必要。本书所得到的所有赞誉都应归功于所有这些人，任何缺陷和不足都属于我们自己。

Kurt Bittner
Ian Spence

2002 年 4 月

目　录

第 I 部分　用例建模入门

第Ⅱ部分 编写和评审用例描述

第Ⅰ部分　用例建模入门

如前所述，用例是一种简单而功能强大的工具，用于表达系统的功能性需求或行为。后文将描述用例的基本概念、它们的基本结构以及格式和内容。最后，介绍一种用例建模的极好方式：工作室，作为第Ⅰ部分的结束。

这些章节的意图是通过连续展示更多的细节（就像剥洋葱皮一样），逐渐向大家介绍基本概念。前几章提供了一些很基本、但是很重要的信息，后面的几章将以这些重要信息为依据。因此，使用用例的人（即使是那些偶而使用用例的人）都会发现前几章很有用，编写用例的人则会完整阅读第1～5章。

第Ⅰ部分的目的就是使我们能够确定参与者和用例、给出简要说明，并且概括和描述用例。第Ⅱ部分在处理编写用例的实践问题方面提供了更多的信息，这些问题涉及到编写用例描述、构造用例模型、管理用例描述中的细节，以及处理编写用例过程中常见的复杂问题。

无需多言，我们开始吧！

第 1 章　用例建模简介

本章的目的是介绍用例建模以及使用用例的原因。其中简要概括了用例建模中所使用的基本概念，描述了这些概念与传统需求捕获技术相关联的方式，并介绍了为何用例可以提供一种捕获和理解系统行为的高级方式。目的就在于为那些不熟悉用例建模的读者提供对用例方法的简要概括，让他们不会被太多细节所困扰。第 2 章“用例建模基础”将更为深入地介绍这些概念。

1.1　参与者和用例

用例建模背后的基本思路非常简单：为了深入了解系统必须做的事情，首先要集中研究谁（或什么）将使用用例，或者用例将会使用什么。在这之后，要看看系统需要为那些用户做什么才能使这些事情有意义。

用例模型包括以下组成部分：

- 参与者——参与者代表了以某种方式与系统交互的人或事；根据定义，它们是在系统之外的。我们将注意力放在参与者上，是为了确保系统可以做出有意义的事情。参与者有一个名称和简短的描述，而且参与者要与它们交互的用例相关联。
- 用例——用例代表了系统为其参与者所执行的有价值的操作。用例并不是功能或特性，也不能进行分解。用例具有一个名称和简要的描述。用例还拥有一些详细描述，这些描述从根本上阐述了参与者如何使用系统来做他们认为重要的事情，以及系统为满足这些需要而做的事情。

用于描述系统的所有参与者和用例的组合就称为系统的用例模型。这也是用例模型的基本概念。但这并不全面——否则这本书就没有什么好写的了。使用用例的微妙之处在于编写用例描述。

1.2　用　例　图

参与者和用例可以在用例图中描绘出来。参与者用简笔人物画来表示，用例使用椭圆来表示。箭头（代表关系）用于连接参与者和与之交互的用例。箭头的头部表示交互的发起者。图 1-1 显示了一个非常简单的电话系统，其中包含一些参与者和用例。该图的目的是总结系统将要做的事情。这个图并没有真正地描述出系统，将用例图误认为是完整的用例模型是许多团队经常犯的错误。图中提供了一个总结，但是作例描述的大部分内容是在

与用例关联的文档中以文本的形式给出的。这些用例描述提供了用例中所发生情况的完整故事。因此，对于用例模型中的每个用例来说，都会有一个文档用于描述参与者与系统如何相互协作来实现用例所提出目标。在本书中，当提到用例时，通常就表示用例的全部，其中包括图标表示、它的关系，而且包括最重要的部分——详细描述。

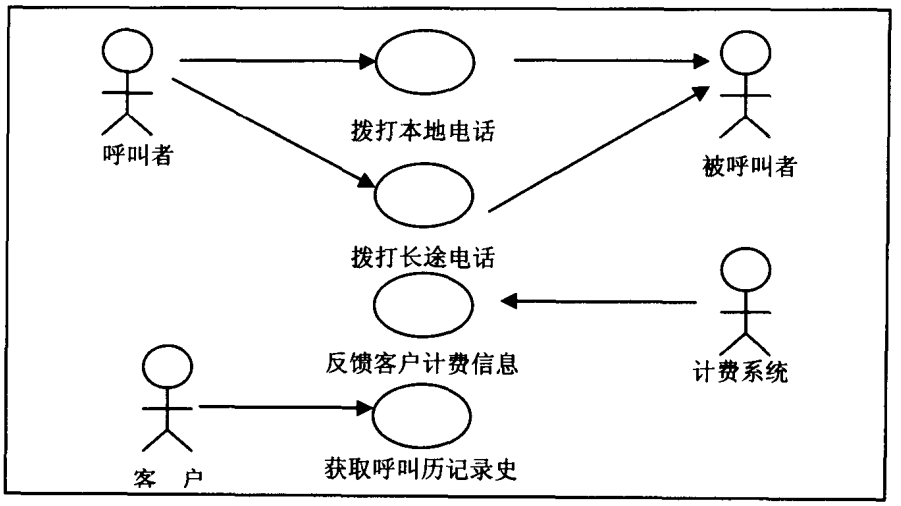

图 1-1 简单的电话系统

这就是用例图。用例可以帮助我们将注意力放在创建系统过程中最重要和最基本的事情上，该系统可以完成有价值的工作。对于系统功能的描述主要采用文本形式来记录；用例图可以作为系统行为的一种概括和总结。

1.3 用例和需求间的关系

用例主要是一种用于表达系统需求的方式，而主要表现的是系统的行为方面。为了理解其中的含义，需要观察需求管理更广泛的环境。需求管理的目的就是在系统应该做什么方面，为客户与其它涉众之间建立和维护一种共识。通常，这种共识要以某种需求规范的形式来记录。

1.3.1 需求类型

需求[①]描述了系统必须遵循的某种条件或能力；需求可以直接从涉众或用户需要中获得，也可以在合同、标准、规范或其他正式制订的文档中说明。有时，表达不同种类的需

① UML（the Unified Modeling Language，统一建模语言）是由对象管理组织（Object Management Group，简称 OMG——参见 *www.omg.org/uml*）提供的一种软件描述标准，它将需求定义为“系统所需的特性、属性或行为”。

求是很有用处的：

- **需要**：涉众认为系统需要完成的事情；涉众需要解决的问题。理解需要的概念固然重要，但是这种说法并不正规，因此需要采用其他一些方式来表达系统的需求。
- **特性**：系统功能的非正式说明，通常用于市场推广和产品定位目的，可以作为系统行为的一组简要说明。尽管在讨论使用临时设置的系统时是有用的，但是在定义系统行为方面却不能提供帮助，因为并没有提供足够精确的术语来对系统进行设计、开发或测试。

特性存在的问题是“特性无处不在”；特性没有精确的定义和/或一致的抽象级别。然而作为系统必须完成功能的简要描述来说，特性还是很有用处的。我们的产品中某一发行版本的特性列表包括：

- **讨论组**——在讨论组中，团队成员可以讨论需求并协调它们的定义。
- **多选列表**——在多选列表中，用户可以为需求属性的价值选择多个值。

在特性方面要注意的重要事情就是这两个特性处于完全不同的抽象级别上。关于特性的关键问题是它突出了系统功能的某些领域，而这些领域在当时对系统的用户是非常重要的。下一个系统发行版本的特性（我们要重点说明的事物）是完全不同的。

既然特性不能用于精确定义系统的功能，那就还需要其他的事物来捕捉系统所需的功能。这就需要使用：

- **软件需求**——系统必须遵循的条件和功能的单独陈述。

每个软件需求都是系统外部可观察行为的规范；例如系统的输入、系统的输出、系统的功能（输入与输出的映射以及它们的各种组合）、系统的属性或系统环境的属性。软件需求作为指定用户或另一个系统的代表软件所做的工作。这些都是详细且明确的需求，足以明确地指导系统的实现和测试工作。

软件需求规范可以采用各种形式来表达。其中最常用的一种形式是使用声明式陈述。在简单电话系统软件需求的示例中可能包括以下内容：

- 在 95%的情况中，从拨号完成到呼叫设备的响铃之间的响应时间应该小于 0.5 秒。
- 拨号错误和连接错误应该报告给用户应使用与话机国家代码相应的主要语言。
- 拨号者或接听者其中一方挂断电话时，系统应该终止呼叫。

在 *Managing software Requirement:A Unified Approach* 一书中，Leffingwell 和 Widrig 使用了图 1-2 中的图形化表示法，来展示不同的需求类型以及它们与问题和解决方案域之间的关系。问题域与解决方案域分隔开表示需求规范的主题是解决方案，而不是问题。需要代表了对用户和其他涉众需要的理解，这些用户和涉众将受到方案的影响。这些是由方案直接解决的问题的一些方面。金字塔的形状反映了需求的相对数量：一些需要可以产生一些特性，而这些特性反过来又要由更多的需求来定义。三类需求之间的这种关系要使用

具有可追踪性的关系来表示。可追踪性是双向的，因为必须在捕获无约束的涉众需要和请求的特性与构建一个符合这些需要[①]的系统的可行性之间保持一种平衡。

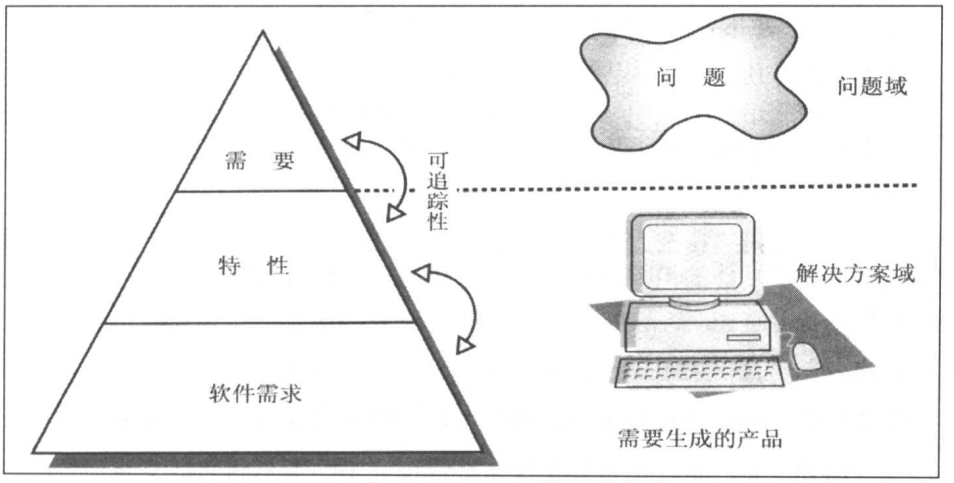

图 1-2 需求类型和可追踪性

需要以非正规的方式表现出涉众需要系统所表现出来的特点，特性是用于表达一个系统（或系统的某个版本）可提供服务的非正规方式，而软件需求表示了系统必须要做的事情。在将需要分解为特性并进而分解为需求的过程中并不存在层次关系。实际上，这些概念在很大程度上是交叉的，所表达的是不同用户对系统的不同的观点。提出需要和特性的概念主要是为用例建模方法的应用建立一种环境；本书的大部分内容都是采用用例形式表达的需求。

1.3.2 功能性需求和非功能性需求

需求有时可以分为两类：

（1）功能性需求（定义系统所需的行为）。

（2）非功能性需求（定义了系统必须符合的其他品质或约束）

功能性需求是系统必须能够实现的那些动作，而不需要在条件中加入物理约束。功能性需求指定了系统的输入和输出行为。非功能性需求指定了系统必须具有的其他品质，如那些与系统的可用性、可靠性、性能和可支持性相关的品质。[②]许多需求都是非功能性的，

① 需要、特性以及用例之间的关系将在第 3 章“建立构想”中给出。

② 记住这些需求类别的其中一种方式就是使用 FURPS+模型（参见 Grady 编著的 *Practical Software Metrics for Project Management and Process Improvement* 一书，Prentice Hall 于 1992 年出版），该模型使用了 FURPS 来描述需求的主要类别：功能性、可用性、可靠性、性能和可支持性。FURPS+中的“+”是一个提示符，表示还要考虑其他一些需求，如设计约束、实现、接口和物理需求。