HZ BOOKS
华章科技

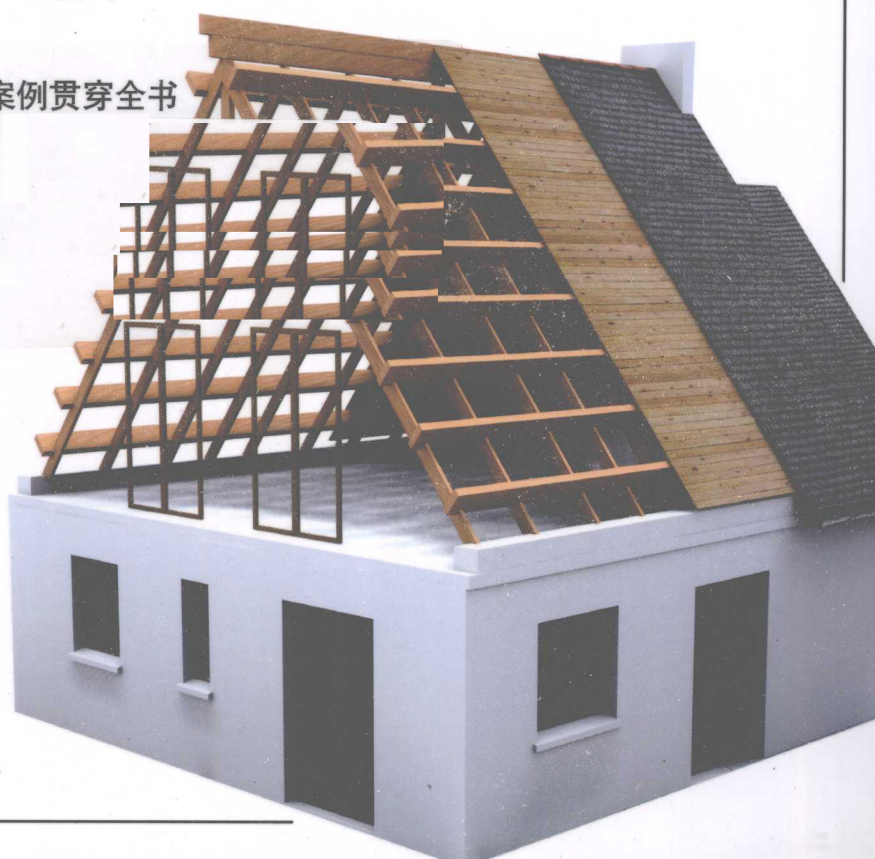
华章原创精品

中国台湾知名UML技术专家邱郁惠老师新作
为UML和OOAD的初学者解开终极困惑

UML和OOAD快速入门

邱郁惠 编著

- 类图、用例图、序列图
- 事务模式、用例描述、BCE模式
- 边做边学，实战演练
- “酒店联合订房系统”案例贯穿全书
- 提供配套系列PPT下载

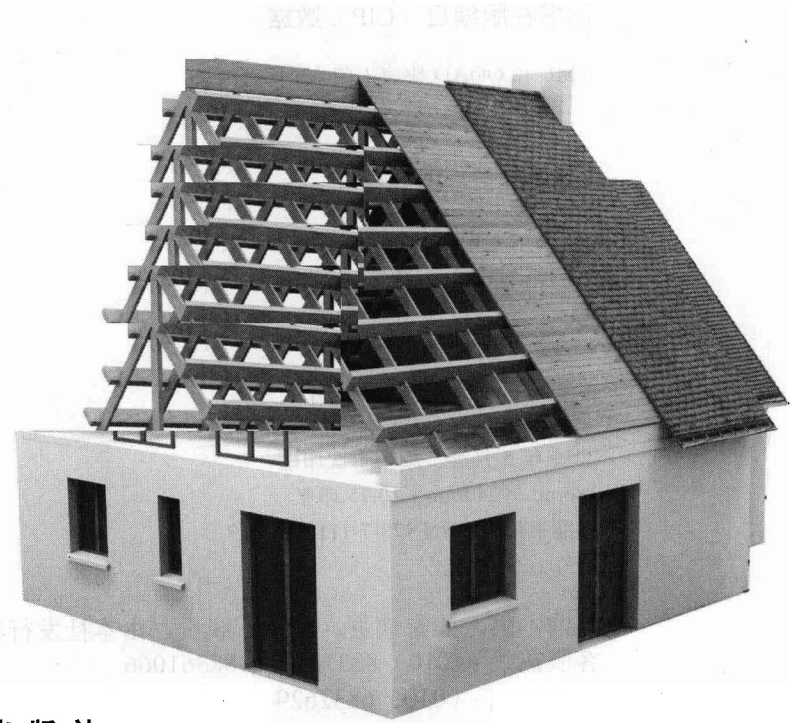


机械工业出版社
China Machine Press

华章原创精品

UML和OOAD快速入门

邱郁惠 编著



机械工业出版社
China Machine Press

本书是 UML 和 OOAD 的初学者指南。

本书介绍了分析师必学的 3 种 UML 图：类图、用例图和序列图，以及 3 种最实用、最常用的 OOAD 概念和工具：事务模式、用例描述、BCE 模式。通过贯穿全书的“酒店联合订房系统”案例，展示了这些图和 OOAD 工具的实际应用。

本书适合 UML 和 OOAD 的初学者以及对 UML 和 OOAD 感兴趣的开发人员，可以帮助他们掌握 UML 和 OOAD 的基本技能和技巧。

图书在版编目（CIP）数据

UML 和 OOAD 快速入门 / 邱郁惠编著 . —北京：机械工业出版社，2010.5

ISBN 978-7-111-30444-9

Ⅰ . U…　Ⅱ . 邱…　Ⅲ . 面向对象语言 - 程序设计　Ⅳ .TP312

中国版本图书馆 CIP 数据核字（2010）第 070786 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：李东震
北京市荣盛彩色印刷有限公司印刷
2010 年 5 月第 1 版第 1 次印刷
186mm×240mm · 10.75 印张
标准书号：ISBN 978-7-111-30444-9
定价：29.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：(010) 88378991；88361066
购书热线：(010) 68326294；88379649；68995259
投稿热线：(010) 88379604
读者信箱：hzjsj@hzbook.com

谨以本书献给我的父母、先生和小孩

——没有他（她）们就没有我

前　言

本书使用的技术

E 时代讲求快速、轻薄，在系统开发上也是如此。可是 UML 2 有 14 种图，并不符合轻薄的需求，全部用起来也不快速。当然，UML 希望自己可以用在各种系统开发中，所以有理由厚重。但是，我们讲求快速入门，因此只选用其中必用的三种图：类图（class diagram）、用例图（use case diagram）和序列图（sequence diagram）。

不过，UML 这三种图不太够，所以我们搭配了其他技术，如下：

1. 事务模式（transaction patterns）。应用事务模式，快速绘制出类图。
2. 用例描述（use case description）。针对用例图中的每个用例，以文字方式描述用例的执行流程。
3. BCE 模式（Boundary-Control-Entity patterns）。应用 BCE 模式，帮助绘制出序列图。

此外，UML 本身只是一种单纯的图形语言，并不包含分析设计步骤，所以本书提出了一系列联系紧密的分析设计步骤。首先，由分析师交付一套分析阶段的类图、用例图和序列图的文件给设计师；接着，设计师根据这套分析文件，添加与实际技术有关的设计内容，生成另一套类图、用例图和序列图的设计文件给程序员。

酒店联合订房系统

本书以“酒店联合订房系统”为主要范例，在讲述任何概念时，如果没有特别说明的话，都以这个范例为主。联合订房系统的服务非常明确，会员可以上网向多家酒店订房。成为会员的访客是这个系统的主要用户。为了简化这个范例，我们剔除了后台的管理机制，也就是说，这个系统并没有包含后台的管理功能。

如何阅读本书

全书共分 6 章，前 3 章讲分析，后 3 章讲设计。如果您是分析师，为求快速、省时，可以不读后 3 章的设计部分。当然，在时间允许的情况下，还是建议分析师阅读后 3 章的设计部分，这样会更懂得如何跟设计师沟通。

但是，您要是设计师的话，一定得阅读前 3 章的分析章节，因为分析师也需要学习关于 UML、事务模式和 BCE 模式的概念，所以这些会在分析部分先讲述，您要是跳过前 3 章的话，可能会有点不明就里。

感谢李强、关志兴、王建勇、毛立涛等在本书简体版出版过程中所做的大量工作。

目　录

第1章　类　图

1.1　概述

类图（class diagram）用来表示系统内部的静态结构（static structure）。具体来说，开发人员可以通过类图的设计，将数以万行的程序代码分门别类，以构成系统内部的静态结构。

过去，开发人员在写程序时，需要分模块（module）、定功能（function）、定义变量，这些动作在面向对象（Object-Oriented）技术中，一样都没少。

现在，观念上有两个显著的改变：

1. 新术语。模块变成类（class）、功能变成操作（operation）、变量变成属性（attribute）。新术语并不是旧酒换新瓶，而是在分类、定操作、定义属性基础之上，有新的划分方法。

2. 新的划分方法。以前的做法是从功能的角度，把大功能、大流程分成数个模块；再把功能模块分成小功能、小流程，定出功能；然后在编写功能时，定义所需的变量。新的分法是，拿用户的领域术语当类，然后确定相关的操作和属性，最后将其封装在同一个类中。

所以，再回过头来看，系统的内部结构是由一个个类所组成的，类内部有操作和属性，类和类之间有静态关系（static relationship）。由于类里头同时包含了静态数据（属性），数据之间会有关联的需要，这种以数据为主的关联，即为“静态关系”。也就是说，类图不仅规范了程序代码，其实还同时规范了数据库的数据结构。

在 UML 中，类图的元素相当繁杂，分析师当然不需要全学，所以在接下来的 1.2 节中，我们仅谈论分析师必学的元素。

1.2　分析师必学元素

1.2.1　类

想象一下，将系统数以万行的程序代码划分为一个个的区块，每一个区块即为一个类。每一个类中，包含有操作和属性；操作内部放置与逻辑运算相关的程序代码，属性则是所需的局部变量。

分析师不能自己随意定义类，必须寻找领域术语作为类名称。比如，一谈到订房，我们就会想到打算预订什么样的房间，这里会找到两个领域概念：

1. 房间。真正住进去，特定房号的房间。

2. 房型。顾客在订房时，通常是预订某个房型的房间。

类的图标为矩形，通常矩形内部会分为三格：顶格放置类名称、中格放置属性、底格放置操作，如图 1-1 所示。说明如下：

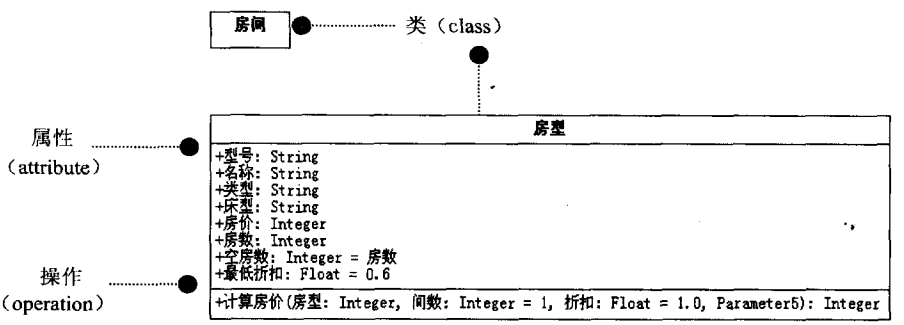


图1-1 类

- 属性。属性是数据项，所以需要指定它的类型（type），属性名称后接冒号隔开类型。如果，属性有默认值的话，也可以在类型后面加上等号，然后标出默认值。
- 操作。操作名称后面以小括号括起输入参数，然后可以在小括号后接冒号，标出返回参数的类型。
- 可见性。特别注意到，属性名称前面有个加号、操作名称前面也有个加号，这个符号称为“可见性（visibility）”。由于属性和操作封装在类中，所以需要使用可见性来标示它们的访问等级。UML 2 设置了 4 种可见性，分析师只要掌握“私有（private）”和“公有（public）”两种就可以了。
- 私有。属性的可见性通常会设成私有等级，除了所属类内部的操作可以直接访问之外，其他类内部的操作不可以访问私有等级的属性。
- 公有。操作的可见性通常设为公有等级，不仅所属类内部的其他操作可以调用（call），其他类内部的操作也可以调用公有等级的操作。

1.2.2 关联

很多分析师都有设计“实体关系图（Entity-Relationship Diagram，ERD）”的经验，我们会在表（table）之间建立关系，这样才能关联不同表内的数据。

比较一下，表跟类最大的区别在于，表只包含数据，但是类同时包含了数据（属性）和操作。也就是说，类其实具备表的静态结构特性，但是又多了一份动态行为特性。至于原先在表之前的关系（relationship），对应到类之间，则称为“关联（association）”。

分析师可以借用以前实体关系图的概念，来认识类图，简单对照如表 1-1 所示。虽然，记录的概念可以对应到对象，但只能对应到半个对象，因为一个对象同时含有属性值和操作，但是一条记录只含有字段值。

表 1-1　实体关系图与类图

实体关系图（关系型数据库）	类图（面向对象）
表（table）	类（class）
记录（record）	对象（object）
字段（field）	属性（attribute）
无	操作（operation）
关系（relationship）	关联（association）

表之间的关系可分为一对一关系、一对多关系、多对多关系，用来表示一条记录能够关联到另一个表中的几条记录。类之间的关联中，同样也这样的概念，叫做“多重性（multiplicity）”，用来表示一个对象能够结合到另一个类中的几个对象。

关联的图标是实线，两个结合端点可以标示多重性。多重性的下限为 0，上限是无限大（*），下限标示在前面，上限标示在后面，两个数字之间使用两个点（..）隔开。如图 1-2 中的例子，表示一个房型可能连接一到多个房间，而一个房间则一定被限定连接到某一个房型。

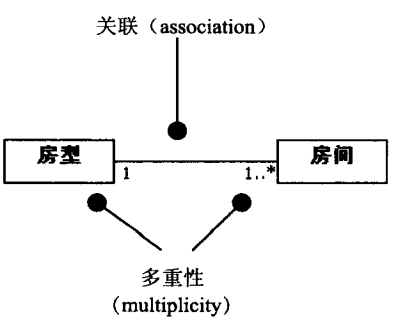


图1-2　关联

在多对多的处理上，也参照关系型数据库的经验，由于处理起来太复杂，所以通常会将多对多拆解成两个一对多的结构。比方说，一次订房可能预订多种房型，而每一种房型也可能跟多个不同的订房事务有关，两者之间形成多对多的关联，如图 1-3 所示。

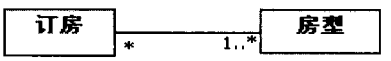


图1-3 多对多的多重性

因为多对多不好处理，所以我们将图 1-3 拆解成两个一对多的关联，如图 1-4 所示。一次订房包含多个订房明细，每一个订房明细则隶属于某一次的订房事务中。再者，一个订房明细会连接到一个房型，每一个房型可能出现在多个订房明细中。

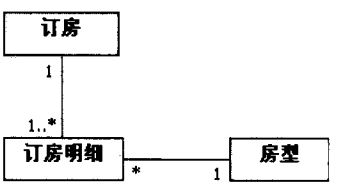


图1-4 两个一对多的多重性

1.2.3 组合关系

再以图 1-4 为例，仔细比较订房－订房明细以及订房明细－房型两者之间的关系，不难发现，订房－订房明细之间的关系比较特别。订房－订房明细之间的关系有一种“整体－部分（whole-part）”的强烈关系，一旦代表整体的订房对象删除的话，底下所连接的订房明细对象都应该一并删除才对。但是，订房明细－房型之间则没有这种强烈的所有关系。

如果分析师要表达两种对象之间有这种强烈的所有权，可以在整体端标示实心菱形，未标示的则代表部分端，如图 1-5 所示。具备强烈整体－部分的关联，特别地称为“组合关系（composition relationship）”。不过，组合关系很容易遭到滥用。整体－部分在领域概念上必须是不可分割或者难以分割的。

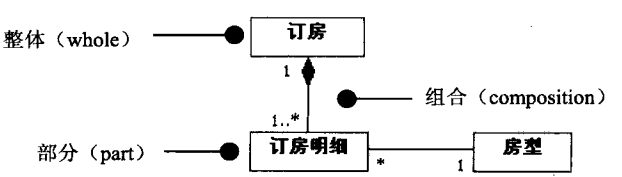


图1-5 组合关系

1.3 事务模式

领域概念非常多，建议分析师应用“事务模式”迅速勾勒出类图的雏型。事务模式由知名的 OOAD 大师 Peter Coad 所提出，您可以在网络上查到许多相关资料，主要出处为《Object Models: Strategies, Patterns, & Applications》一书。

不过，我们在此处的引用上，多了一些限制，所以使用上会比 Peter Coad 所提出的事务模式更狭隘一些，但是并没有改变 Peter Coad 的原意。我在《系统分析师 UML 实务手册》和《系统分析师 UML 用例实战》[⊖]这两本书中，特别是后者中，都详细解释过事务模式，有兴趣的话，推荐您找来读一读。

1.3.1 事务与人、地、物

顾名思义，事务模式强调以“事务（Transaction）”为中心，串起跟事务相关的事务明细（TransactionLineItem）、涉众（Participant）、地点（Place）、物品（Item），如图 1-6 所示。

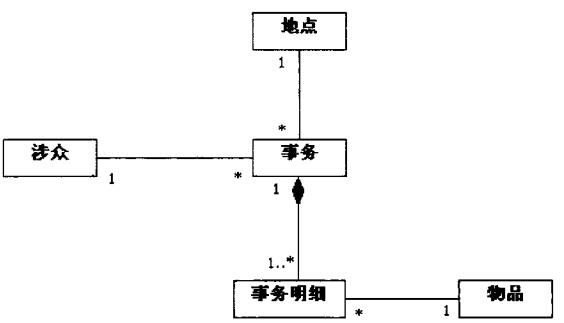


图1-6　事务模式（核心）

不过，此处对“事务”的定义比较广义，并不局限于一般的商业交易，系统中所有“必须记录的事件”都是我们的候选事务。因此，事务可能是一个短暂的时间片刻，也可能是一小段时间。

所以，应用图 1-6 的事务模式，分析师大概可以很快得出如图 1-7 所示的类图雏型。

⊖　这两本书已由机械工业出版社出版，书号为 ISBN 978-7-111-23738-9 和 ISBN 978-7-111-28576-2。——编辑注

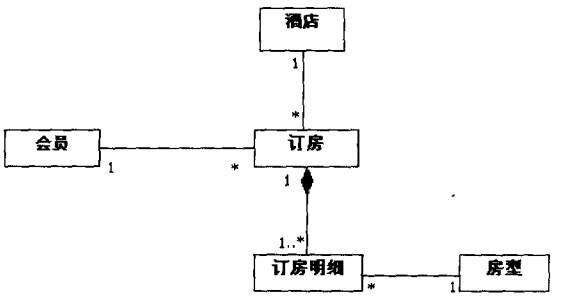


图1-7 应用事务模式

1.3.2 物品与特定物品

事务涉及的物品概念，可以细分成两种：一种是具体的、特定的物品，另一种是针对一群同种类特定物品的描述或分类。在事务模式中，则将这两个概念分为“特定物品（SpecificItem）”和“物品（Item）”，如图 1-8 所示。

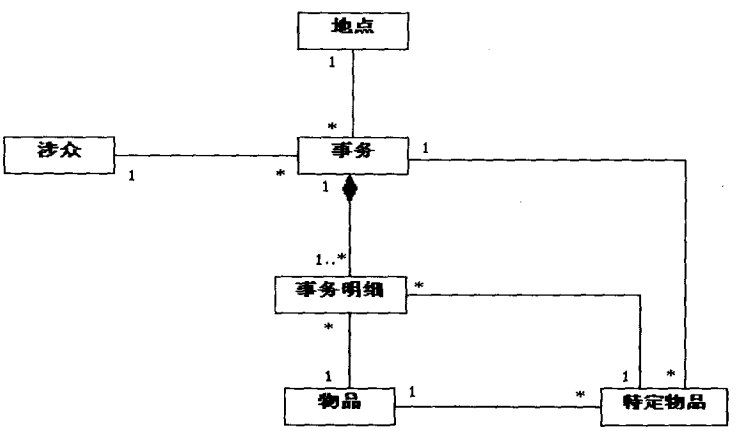


图1-8 物品与特定物品

其实，我们在前面提到的“房型－房间”的概念，就是“物品－特定物品”的应用，如图 1-9 所示。不过，在订房的范例中，我们目前还无法确定是否可以直接应用“特定物品－事务明细”以及“特定物品－事务”。分析师要是遇到这种未确定的情况，可以标上像记事贴一样的“注释（comment）”来提醒自己留意。

之所以无法确定图 1-9 中“房间－订房明细”以及“房间－订房”这两条关联，最主要原因在于，这两条关联都不是单纯的一对多结合，而是如图 1-10 所示的多对多结合。此处，我们先保留多对多结合，稍后再来讨论细节。

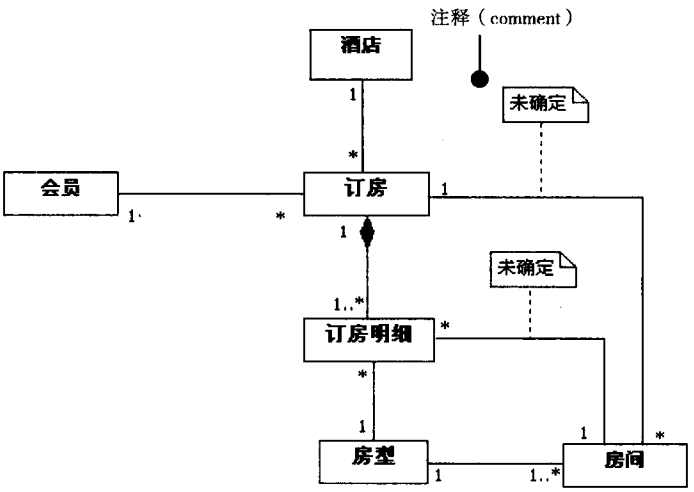


图1-9　应用“物品—特定物品”

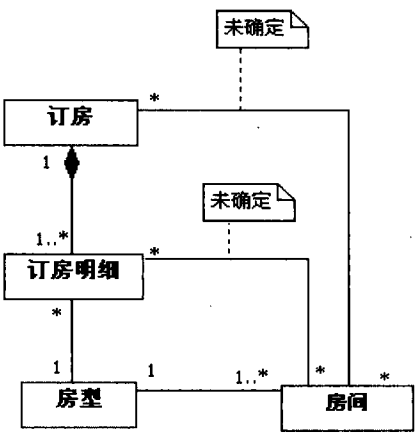


图1-10　多对多的复杂状况

1.3.3　后续事务

事务本身含有时间因素，所以如果拉出一条时间轴的话，会看到事务之后可能有“后续事务（subsquentTransaction）”，如图 1-11 所示。

想想看，订房成功之后，后续会发生哪些必须记录的重要事件？我第一个想到的后续事务是“入住（check in）”，如图 1-12 所示。

应用事务模式不仅可以帮助分析师快速作出类图雏型，更重要的是，可以协助分析师使用比较系统性的方法去理解领域概念。比方说，应用事务模式到目前为止，好像对“订房－

订房明细”以及“入住－入住明细”领域概念一直没有真正理清，分析师可能开始觉得有必要理清这两个概念。

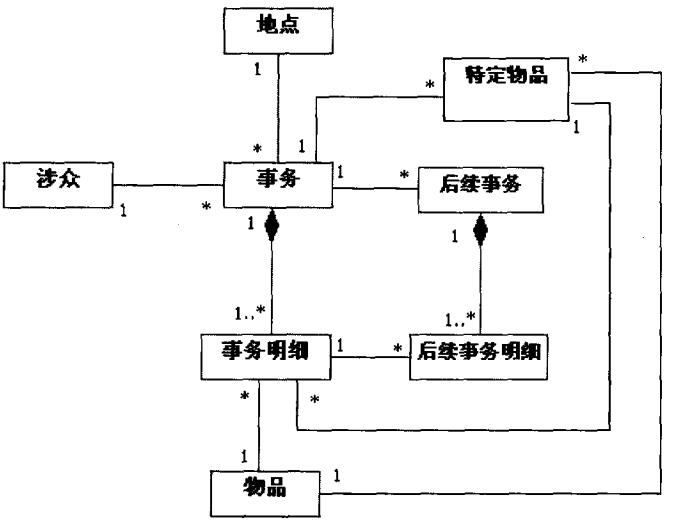


图1-11 后续事务－后续事务明细

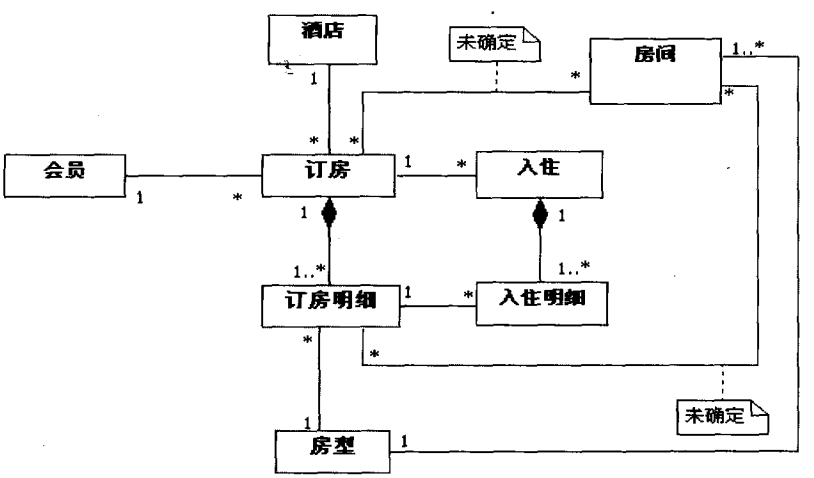


图1-12 应用“后续事务－后续事务明细”

首先，我们先理清订房－订房明细，如图1-13所示，我们做下述规定，让它看起来简单一点：一次订房可以同时订多个房间，但是只能限定是同一个预订日期。假设，这个会员

要一次预订多天的话，必须拆成多个订房事务。

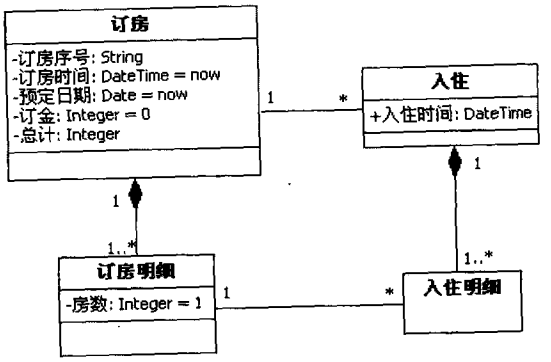


图1-13　订房与入住

再者，我们还规定一次入住事件对应一间房间。假设会员预订 12/20 两个房间的话，那这次订房就会对应到两个入住事件。在这样的规定之下，也就不再需要入住明细了，因此修改成图 1-14 的样子。

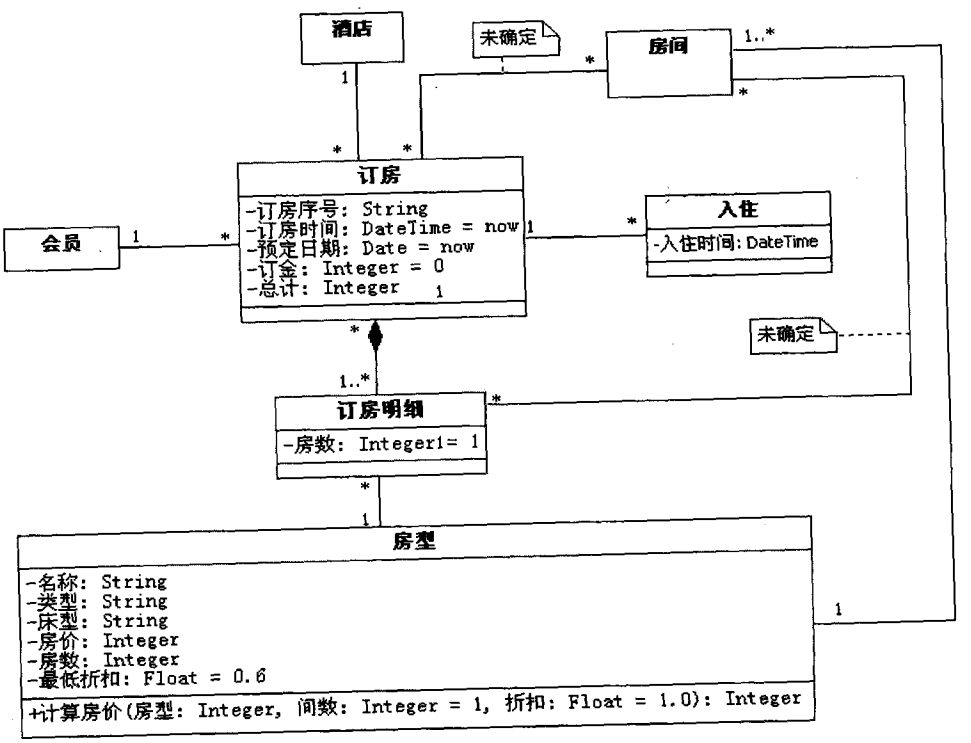


图1-14　删除“入住明细”

1.3.4 参与者与涉众

最后，我们回过头来看事务模式中的涉众的概念。仔细探究起来，涉众其实是一种身份、一种角色，在这个角色背后有一个真正的“参与者（actor）”。“参与者－涉众”之间的关系就像是“演员－角色”一样，如图 1-15 所示。

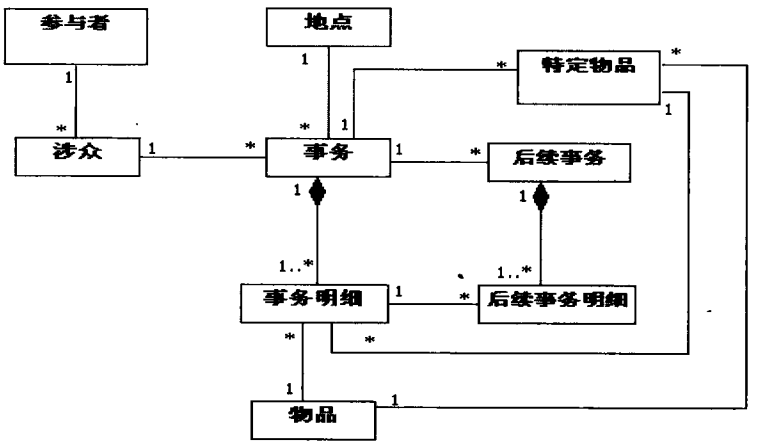


图1-15 参与者-涉众

在订房系统中，只有具备会员身份的人才能够订房，所以“个人－会员”就对应到了“参与者－涉众”，如图 1-16 所示。不过，我们还不确定一个人是否可以申请多个会员身份，所以标上了未确定的注释。

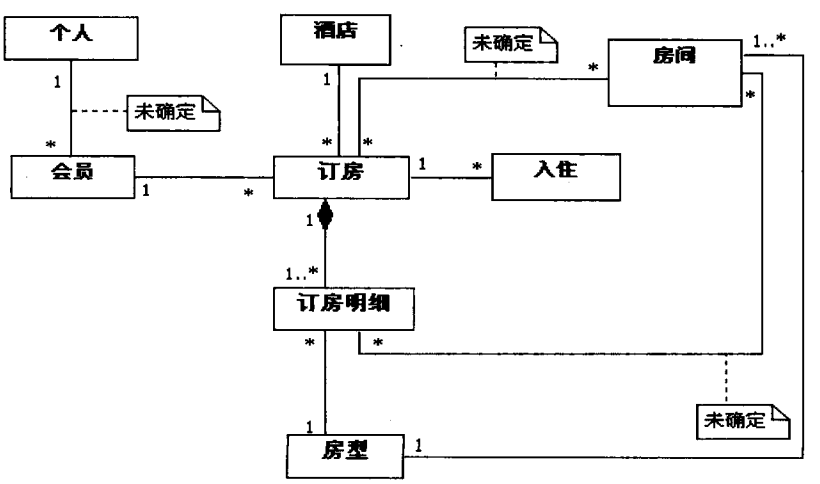


图1-16 应用“参与者-涉众”