第4章 用例图

教学要求

- 掌握：用例图定义、用例图的标记符组件，以及如何建立用例图模型。
- 理解：用例图建模的原因、泛化技术，以及包含与扩展关系。
- 了解：描述用例的方法。

4.1 用例图概述

4.1.1 用例建模的目的

在早期面向过程的软件开发方法中，人们致力于用计算机能够理解的逻辑来描述和表达待解决的问题及其具体的解决过程。于是数据结构、算法成为了面向过程问题求解的核心组成。利用这种开发方法可以精确、完备地描述具体的操作过程。

例如，我们可以容易地利用面向过程的求解方法解决下面这个问题：有一张信用卡，卡上已经产生应还金额 5000 元，假定月利息为 2%，你一直不还款，那么在多少个月之后，这张卡的应还金额会超过 10000 元？

然而如果我们将这个范围扩大，要处理一个有关银行日常业务的问题，那么所有的资金、账目往来，包括存款、取款、贷款、还款和这些操作所处理的数据，如金额、账号、日期等问题都需要考虑，这时利用面向过程的软件开发方法就很难把这个包含了多个相互关联的过程的复杂系统表述清楚了。

为了符合人们日常的思维习惯，降低、分解问题的难度和复杂性，提高整个求解过程的可控制性、可监测性和可维护性，人们提出了一种全新的程序设计思路和观察、表述、处理问题的方法，利用“系统”的观点来分析问题、解决问题。这时人们关心的就不仅仅是孤立的单个过程，而是孕育所有这个过程的母体系统，它使用计算机逻辑来模拟描述系统本身，包括系统的组成、关系、系统的各种可能状态、系统中可能产生的过程和过程引起的切换。

既然 UML 作为一种建模语言主要是为帮助用户对软件“系统”进行面向对象的描述和建模，并描述整个软件开发从需求分析到实现和测试的全过程的，那么如何在 UML 的背景下表述需求、分析系统、建模软件呢？答案是，可以利用一个容易理解的模型来描述用户如何使用这个系统、系统和客户以及系统和外部系统之间的交互过程，这个模型也就是通常所说的使用 UML 设计新系统

的起始点——用例图。

4.1.2 定义用例图

用例图是有关系统细节的最高形式。它能准确地说明客户对他们要开发的应用程序期望有什么样的功能，同样是一种在系统完成后能使管理机构、用户和其他干系人了解其功能的极好方法。

对于上文所提到的银行日常业务问题，可以利用“系统”的观点，用用例图初步表述成如图 4-1 所示。

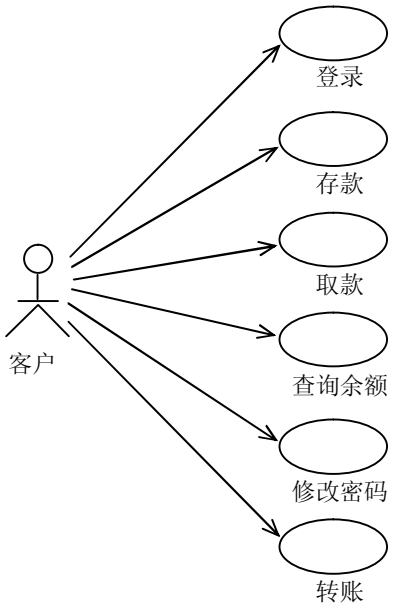


图 4-1 银行日常业务

这个用例图并没有太多内容，但很直观地表述了系统的功能及其边界。但是我们还是要用一些时间来研究每一个组件以及如何建立这样的用例图。

希望通过本章的学习，读者可以完成一个用例模型，包括一个完整的用于导航的高层次的用例图以及对每一个用例的详细描述。这些元素共同组成了一个易懂的、完整的模型。在相关人员评审这个模型来验证目标系统的时候，以及开发小组将这个用例模型作为从工作量估算到设计、测试的整个开发过程的基础的时候，这些元素都是很关键的。

4.1.3 用例图的主要组件

用例图包含 4 个基本组件：参与者（Actor）、用例（Use Case）、关系、系统。

（1）参与者（Actor）。参与者（如图 4-2 所示）是系统外部的一个实体，它以某种方式参与用例的执行过程。参与者通过向系统输入或请求系统输入某些事件来触发系统的执行，所以参与者可以是人，可以是事物，也可以是时间、气压、温度等环境因素或其他系统等。它在系统之外，与

系统直接交互。用一个群体概念给参与者命名，反映该参与者的身份和行为（如客户、管理员等）。

（2）用例。用例代表系统的某项完整的功能，是动作步骤的集合。系统的功能是通过参与者使用用例来实现的。在 UML 中，用例用一个椭圆来表示，用例的名字可以书写在椭圆的下方或中间，如图 4-3 所示。

图 4-2　参与者的符号

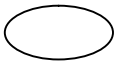


图 4-3　用例的符号

（3）关系。除了用例和参与者之间的关联关系以外，还可以定义参与者之间的泛化关系，用例之间有包含、扩展和泛化关系。应用这些关系的目的是从系统中抽取出公共行为及其变体。这一部分稍后介绍。

（4）系统。系统指一个软件系统、一项业务、一个商务活动、一台机器等。系统的功能通过用例来表现，换句话说就是所有用例共同构成了整个系统。在 UML 中，系统可以用矩形框来表示（如图 4-4 所示），通常也可以将矩形框省略。

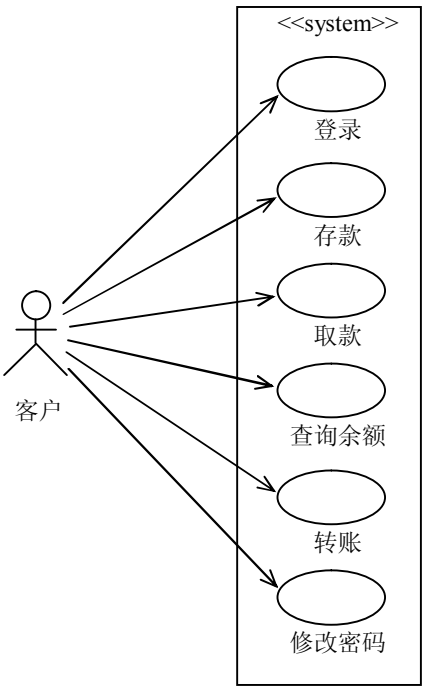


图 4-4　银行对普通客户的日常业务系统

银行业务范围很广，为什么我们并没有提及其他的呢？这里涉及一个确定系统的边界问题。一个系统的基本功能及边界多半是根据前期了解客户的要求、制定的需求分析而明确的。就如图 4-4 所示一样，它只是一个银行对普通客户的日常业务系统。另外，根据系统的大小，一个系统也可能会有自己的子系统。

4.2 识别参与者

4.2.1 捕获需求

设计任何企业级应用程序的第一个步骤都是收集系统需求，它是最终用户、开发者、客户对系统应该做的和能够做的事情达成的协议。从内容上说，除了对应用程序本身功能、性能上的要求之外，还包括描述公司运营方式的业务需求（对复杂系统有时需要首先建立业务模型，再对软件系统建模）、运营要求，以及相关的技术需求等。

收集需求的方式有访谈、问卷调查、实地观察、使用原型、特定群体调查、用户指导等。需求的来源主要是人、各种现有成品（如报表、培训手册、视频记录等）、现有的软件系统或人工系统。

通过各种形式记录下收集到的需求信息，经过整理，从中获取有价值的信息来建立系统模型。

4.2.2 识别参与者

整理需求的第一步是确定参与者，不同的参与者必须以独特的方式来使用这个系统。

下面来看这样一个练习：

- 银行日常业务系统
- 销售跟踪系统
- 选课登记系统

在这些系统里，参与者是谁？怎样识别？这里提供一个识别参与者的思路，可以从以下几个方面来考虑：

（1）谁使用系统的主要功能？

（2）谁改变系统的数据？

（3）谁从系统获取信息？

（4）谁需要系统的支持以完成日常工作任务？

（5）谁负责维护、管理并保持系统正常运行？

（6）系统需要处理哪些硬设备？

（7）系统需要和哪些外部系统交互？

（8）谁对系统运行产生的结果感兴趣？

（9）有无时间、气温等内部或外部条件？

认真考虑这些问题，不难看出：

- 在银行日常业务系统中，客户和出纳可以作为参与者，因为他们使用了系统的主要功能，并且有不同的需求和权限。
- 在销售跟踪系统中，销售人员、客户服务人员和客户可以作为参与者，因为他们有不同的需求和访问方式。

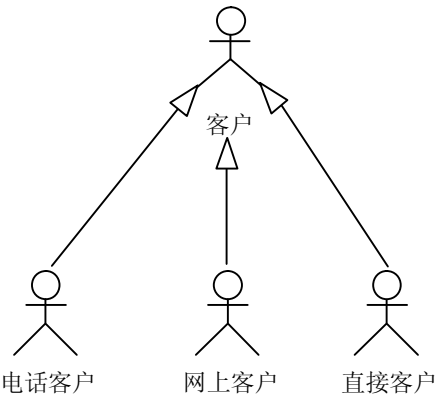
- 在选课登记系统中，学生、教师和管理登记人员可以作为参与者，同样也因为他们有不同的需求、访问方式和权限。

在考虑参与者的同时，也要注意这样一个问题——怎样处理有共同特征的多个参与者。以酒店的订餐预约系统为例，该系统接受客户的电话预订、网上预订和上门预订，那么参与者就有 3 个：电话客户、网上客户和直接客户，如图 4-5（a）所示。

可以看出它们有着共同的行为特征，因此可以抽象为更具一般化的参与者——客户。在用例图中，使用泛化关系来描述多个参与者之间的公共行为，用空心三角箭头指向抽象出来的参与者，如图 4-5（b）所示。

不同的参与者必须是以独特的方式来使用系统，如果三类参与者都与相同的用例相关联（即使用相同的功能），则只需保留客户作为参与者，如图 4-5（c）所示。

（a）订餐预约系统的参与者



（b）泛化后的订餐预约系统的参与者

（c）最终确定的参与者

图 4-5　订餐预约系统的参与者的确定

大家可以练习一下，对于飞机订票系统而言，有几类参与者？他们之间的关系是什么？

4.3　识别用例

4.3.1　识别用例的方法

识别用例的最好方法是从前期所了解的客户需求入手，分析系统的参与者，考虑每个参与者是如何使用系统的，在这个过程中还可能会发现新的参与者，有助于完善系统。

在识别用例的过程中，可以从以下几个方面来考虑：

（1）特定参与者希望系统提供什么功能。

（2）系统是否存储和检索信息，如果是，由哪个参与者触发。

（3）当系统改变状态时，是否通知参与者。

（4）是否存在影响系统的外部事件。

依旧从上文提及的酒店订餐预约系统来看，通过分析发现：除了客户，希望系统提供功能的还有 3 类用户：前台接待、服务员、领班。根据他们不同的需求和权限，通过泛化，将其归为两类参与者，即一般员工和领班。

在识别用例的过程中，通过对上面几个问题的回答，可以得到这样的结论：

- 客户：查询预约。
- 一般员工：查询预约、增加预约、删除预约。
- 领班：调换餐桌、确认记录未到达（即客人在指定时间未到达）、查询预约、增加预约、删除预约。

于是，构建初始用例图如图 4-6 所示。

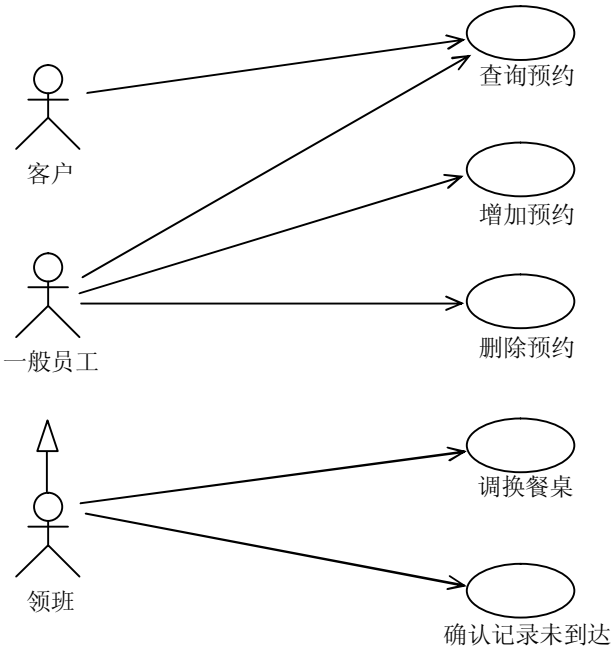


图 4-6　订餐预约系统的初步用例模型

这里只是一个初步确定的用例模型，还有一些问题、细节并没有考虑到，还需要对这些用例进行分解、合并、删除，直到获得一个可靠的用例集。这些在后面的进一步分析中来解决。用例建模的过程就是一个反复迭代和逐步精化的过程，很难一蹴而就。

4.3.2　用例的命名规则

用例名是一个字符串，用例是从用户的角度来描绘系统的功能，因此命名的基本原则是：从

参与者的角度出发进行命名（如使用“登录”而不用“身份验证”），使用动词加宾语的结构，尽量使用行业术语（如使用“报销”、“预支”，而不用“交钱”），示例如图 4-7 所示。

系统中用例太多时需求要适当分组（包），这时可以通过在用例名后面加上双冒号和包名来表示该用例是属于哪个包的。例如图 4-8 中，用例 create software 来源于 development 包。

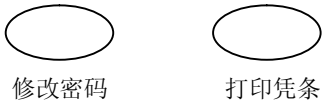


图 4-7　用例命名 A

图 4-8　用例命名 B

4.4　用例间的关系

4.4.1　泛化关系

用例与用例之间也存在着泛化关系，通常用于表示同一业务目的（父用例）的不同技术实现（各个子用例）。

例如，某购物网站为用户提供不同的支付方式，那么“支付”这个复杂用例就可以用泛化关系表示，如图 4-9 所示。

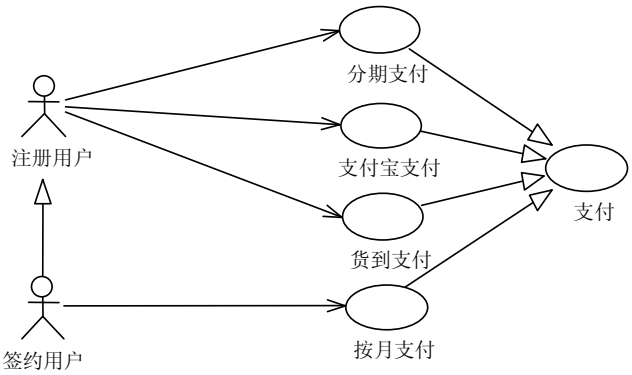


图 4-9　“支付”用例泛化

同样，大家也可以思考一下上文所提及的订餐预约系统，当最终用户提出需要提供多种查询预约方式时，那么“查询预约”这个用例是不是也可以用泛化关系来表示，如图 4-10 所示。

4.4.2　包含关系

首先，在理解包含关系之前，先来看一下图 4-11。

在图 4-11 中，可以看出无论是添加、修改还是删除学生信息，都需要更新数据库信息，这时“更新数据库”就是一个公共的功能模块，可以用虚线箭头加“<<include>>”字样来表示用例之间的包含关系，如图 4-12 所示。

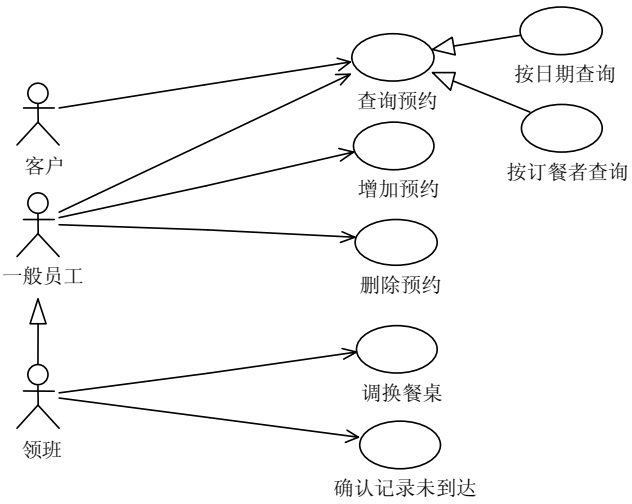


图 4-10 “查询预约”用例泛化

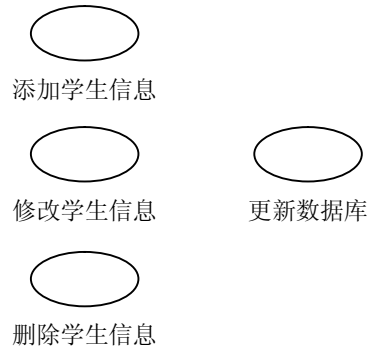


图 4-11 学生个人信息登记系统的用例

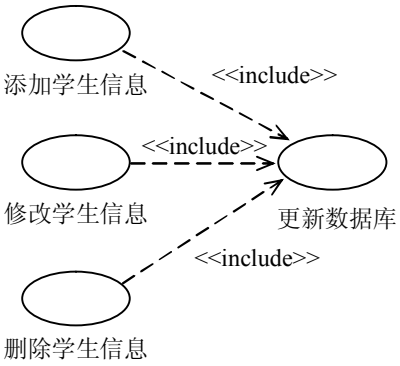


图 4-12 学生个人信息登记系统用例之间的包含关系

在包含关系中，基本用例吸收了被包含的用例的行为，如果没有后者它将是不完整的。

包含关系的划分有两个好处：一是被包含用例被抽取出来，基本用例得以简化；二是可以抽象出公共事件流，实现功能代码的复用。

练习：

由于不同银行之间的 ATM 机存在着一定的差异性，大家可以根据自己的所见所用尝试着修改图 4-13 所示的“ATM 自动取款机系统的用例图”。

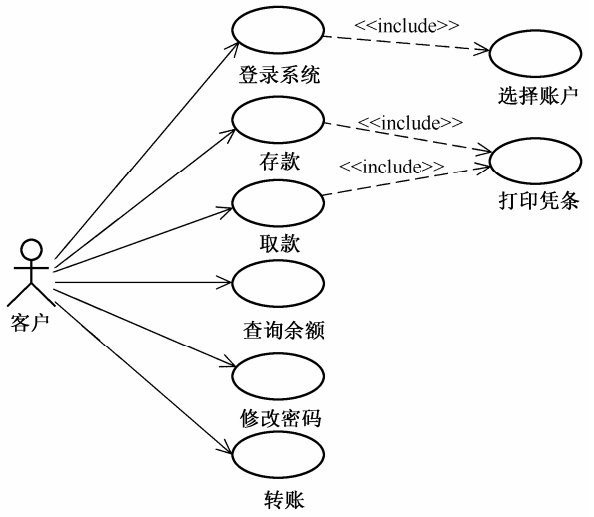


图 4-13　ATM 自动取款机系统

4.4.3　扩展关系

与包含关系极为相似的是扩展关系，如果在完成某个功能的时候有时（偶尔）会执行另一个功能，则用扩展关系来表示。扩展关系表示为虚线箭头加“<<extend>>”字样，箭头指向被扩展的用例。例如，教师在保存成绩的时候，如果有学生成绩不合格将打印补考通知单，可表示成如图 4-14 所示。

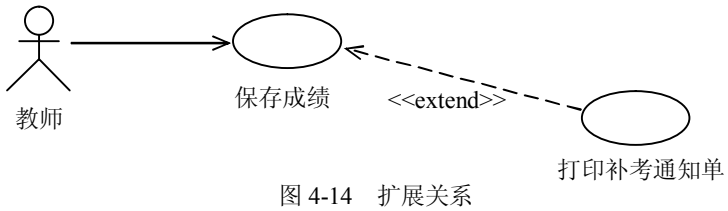


图 4-14　扩展关系

比较图 4-15 和图 4-16 发现，两个用例图在“保存成绩”与“确认成绩”用例之间分别使用了包含关系和扩展关系，这时语意出现了明显的不同，图 4-15 表示教师在保存成绩时必须确认成绩，而图 4-16 表示教师在保存成绩时可能只在某些时候必须确认成绩，例如出现不及格的情况时。

一般情况下，基础用例的执行不会涉及扩展用例，只有在特定的条件发生时，扩展用例才被执行。因此，在系统模型架构时，基础用例的异常处理功能通常用扩展用例表示。

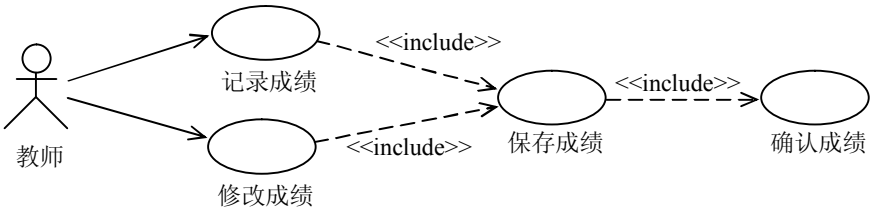


图 4-15　成绩登记系统 A

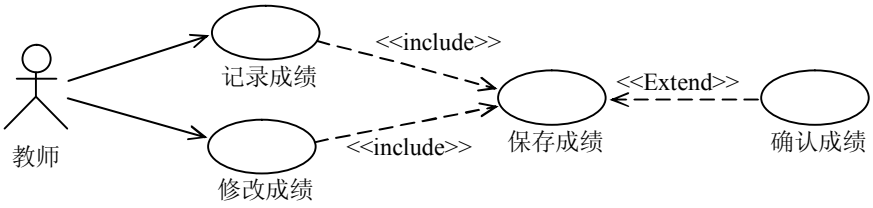


图 4-16　成绩登记系统 B

练习：

表示出以下参与者和用例间的关系：

- 参与者：管理员。
- 用例：记录还书信息、记录借书信息、查询书籍、罚款。

4.5　用例文档

用例图不足以表达更多的细节，对于系统中的每一个用例，还需要了解其详细的执行情况，以便完整地理解创建系统时的设计任务。用例文档为我们提供了一个很好的模板，如图 4-17 所示。

下面再介绍一个简化的用例文档模板。

- 用例编号
- 用例名
- 用例描述
- 参与者
- 前置条件
- 后置条件
- 事件路径
- 扩展点
- 补充说明

事实上，在描述每个用例的事件路径并试图描述前置条件和部署约束时，往往会发现一些其他的问题和事务，从而进一步完善用例模型。下面来阅读两份用例文档，希望大家能从中获得启发。

- **名称**。名称无疑应该表明用户的意图或用例的用途，如“查询书籍”。
- **标识符 [可选]**。唯一**标识符**，如“UC1701”，在项目的其他元素（如类模型）中可以用它来引用这个用例。
- **说明**。概述**用例**的几句话。
- **参与者 [可选]**。与此用例相关的参与者列表。尽管这则信息包含在用例本身中，但在没有用例图时，它有助于增加对该用例的理解。
- **状态 [可选]**。指示用例的状态，通常为以下几种之一：进行中、等待审查、通过审查或未通过审查。
- **频率**。参与者访问此用例的频率。这是一个自由式问题，如用户每天访问一次或每月访问一次。
- **前置条件**。一个条件列表，如果其中包含条件，则这些条件必须在访问用例之前得到满足。
- **后置条件**。一个条件列表，如果其中包含条件，则这些条件将在用例成功完成以后得到满足。
- **被扩展的用例 [可选]**。此用例所扩展的用例（如果存在）。扩展关联是一种广义关系，其中扩展用例接续基用例的行为。这是通过扩展用例向基用例的操作序列中插入附加的操作序列来实现的。这总是使用带有 <<extend>> 的用例关联来建模的。
- **被包含的用例 [可选]**。此用例所包含用例的列表。包含关联是一种广义关系，它表明对处于另一个用例**之中**的用例所描述的行为的包含关系。这总是使用带有 <<include>> 的用例关联来建模的。也称为使用或具有（Has-a）关系。
- **假设 [可选]**。对编写此用例时所创建的域的任何重要假设。你应该在一定的时候检验这些假设，或者将它们变为决策的一部分，或者将它们添加到操作的基本流程或可选流程中。
- **基本操作流程**。参与者在用例中所遵循的主逻辑路径。因为它描述了当各项工作都正常进行时用例的工作**方式**，所以通常称其为适当路径（Happy Path）或主路径（Main Path）。
- **可选操作流程**。用例中很少使用的逻辑路径，那些在变更工作方式、出现异常或发生错误的情况下所遵循的**路径**。
- **修改历史记录 [可选]**。关于用例的修改时间、修改原因和修改人的详细信息。
- **问题 [可选]**。如果存在，则为与此用例的开发相关的问题或操作项目的列表。
- **决策**。关键决策的列表，这些决策通常由你的 SME 作出，并属于用例的内容。将这些决策记录下来对于维护团体记忆库（Group Memory）是相当重要的。

图 4-17　用例文档模板

1．ATM 自动取款机“取款”用例的用例文档示例

用例编号：001

用例名：取款

用例描述：用户使用银联卡在 ATM 机上取款

参与者：用户

前置条件：用户已登录

后置条件：从账户中扣除取款金额，打印或不打印凭条

事件路径：

1．用户选择取款

2．系统要求输入取款金额

3．用户输入取款金额

4．系统验证取款金额

4a 余额不足

4a1 系统显示余额不足

4a2 用例终止

5．系统询问是否打印凭条

6．用户选择不打印凭条

7．系统显示取款成功

2．订餐预约系统“增加预约”用例的用例文档示例

用例编号：002

用例名：增加预约

用例描述：酒店一般员工为顾客增加一次订餐的预约

参与者：一般员工

前置条件：注册用户已登录

后置条件：存储预约信息

事件路径：

1．接待员输入要预约的日期

2．系统显示该日的预约

3．有一张合适的餐桌可以使用，接待员输入顾客的姓名和电话号码、预约的时间、用餐人数和餐桌号

3a 没有合适的餐桌可以使用

3a1 用例终止

4．系统记录并显示该预约

4a 输入的预约人数多于餐桌能容纳的人数

4a1 系统发出一个警告信息，询问用户是否想要继续预约

4a1a 如果回答“否”，用例将不进行预约而终止

4a1b 如果回答“是”，预约将被输入，并附有一个警告标志

在分析用例、书写用例文档时，要注意以下几个问题：

（1）前置条件必须是系统在用例开始前能检测到的。因此，在上面的 ATM 自动取款机“取款”用例中，“用户账户里有足够余额”就不是“取款”用例的前置条件，因为它无法事先被系统检测到。

（2）后置条件是这个用例执行后对系统产生的所有改变。

（3）事件路径书写尽量使用主动句，以参与者或系统为主语，不要涉及软件实现的细节（如选择菜单、单击按钮、修改数据库等）。

（4）事件路径的扩展点一般是由参与者或系统引起的变更而形成的事件流。

练习：

大家尝试一下书写 ATM 自动取款机“登录”用例的用例文档。

4.6　重构系统的用例模型

在整个用例模型的建立过程中，开发者和客户可以达成对系统的初步共识。但用例建模的过程本身就是一个反复迭代和逐步精化的过程，所以在进一步的开发中，还必须对用例进行评估，对用例模型进行不断的修改与完善。

一般从以下几个方面来判断、评估用例的合理性：

（如果你对以下问题都回答“是”，那么这个用例就是合理的；否则，这个用例需要拆分为几个小的用例。）

- 这个用例是否能够带来一个独立的好处？
- 是否可以用简洁的文字来描述这个好处？
- 参与者是否能够仅通过一次会话就完成这个用例？
- 能否想象在一个连贯的测试计划中，这个用例将是一个测试用例？

（如果你对以下问题都回答“是”，那么这个用例就是有效的和独立的；否则，这个用例实际上可能是其他用例的一个部分。）

- 参与者是否得到了明确的信息或者以某种可度量的方式改变系统？
- 执行这个用例之后，参与者是否可以在确定的时间内停止使用这个系统？

另外，描述用例是否使用了技术语言，而非业务语言。技术语言的使用不易于客户的理解。依据这个原则，上文的“订餐预约系统”用例图中的“增加预约”、“删除预约”改为“记录预约”、“取消预约”更为合适。

经过修改、分析、调整，画出最终“订餐预约系统”的用例图如图 4-18 所示。

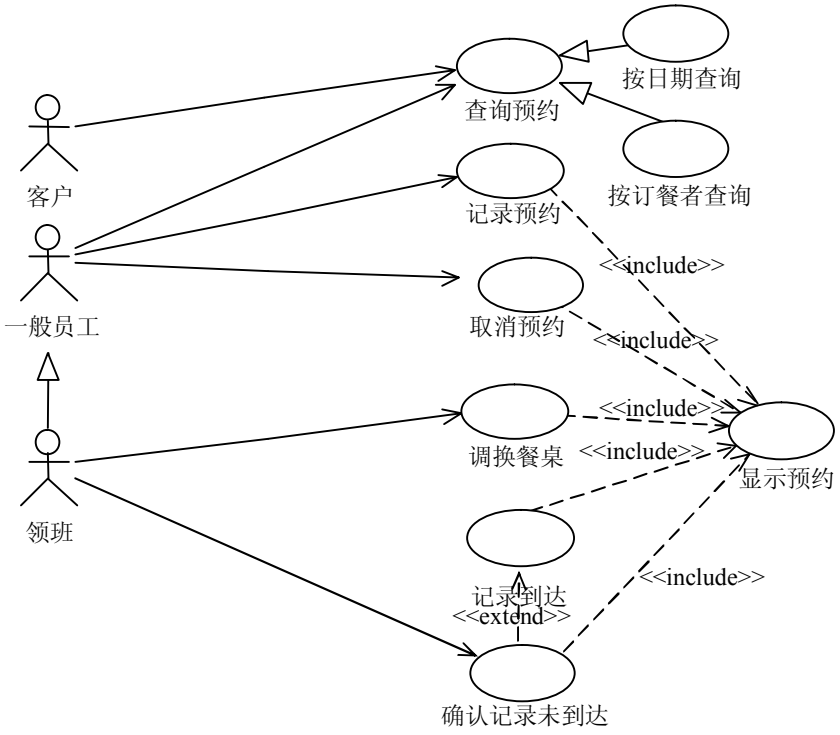


图 4-18　订餐预约系统的最终用例模型

本章小结

用例模型是分析功能需求的一个有力工具。它由用例图和每个用例的文档组成。用例图可以可视化地表达出用例功能，使分析员与用户之间的交流更加容易。在用例图中，用例的表示符号是一个椭圆，参与者的图符是一个直立人形，参与者与用例之间用关联线连接，通常用例都位于表示系统边界的矩形框之中。

用例之间存在各种关系：包含关系用带关键字<<include>>的虚线箭头线表示；扩展关系用带关键字<<extend>>的虚线箭头线表示；还有一种泛化关系，表示一个用例继承了另一个用例的属性和行为。

分析过程开始于和客户交谈，产生系统高层用例图。用例图在分析过程中起着很重要的作用，它能反映系统基本的功能需求。但要创建完整的用例模型，还要对每个高层用例进行细化，建立用例文档。对于复杂的系统可以先画出表达系统整体功能的顶层用例模型，再画出各个功能的用例模型子图。用例模型是后期设计和开发的基础。

习题4

1. 图 4-19 所示是“成绩管理系统”用例模型的一部分，如何处理多个参与者与同样的用例“浏览成绩”相关联的情况？

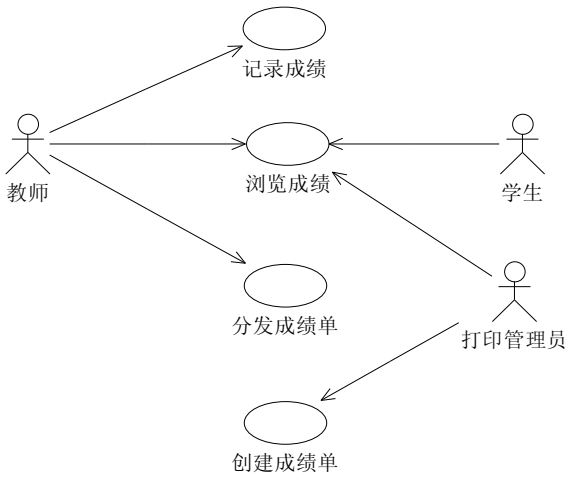


图 4-19 “成绩管理系统”的用例模型

要点提醒：

- 当多个参与者与相同的用例关联时，考虑是否泛化成一个参与者。
- 当多个参与者与同一个用例关联时，考虑是否把该用例特化成不同情况。

究竟使用哪种方案将依赖于实际需求。

2．以下是某销售网点的需求，试分析并画出其用例图。

- 系统允许管理员通过从磁盘加载存货数据来运行存货清单报告。
- 管理员通过从磁盘加载、向磁盘保存数据来更新存货清单。
- 销售员记录正常的销售。
- 电话操作员是处理电话订单的特殊的销售员。
- 任何类型的销售都要更新存货清单。
- 如果交易使用信用卡，那么销售员需要核实信用卡。
- 如果交易使用支票，那么销售员需要核实支票。

要点提醒：

❖ 准确识别参与者、用例，并分析它们之间的关系。