第17章
用　况　图

本章内容：

- 对系统的语境建模
- 对系统的需求建模
- 正向工程与逆向工程

UML中的用况图是对系统的动态方面建模的5种图之一（对系统的动态方面建模的其他4种图是活动图、状态图、顺序图和协作图）。用况图主要用于对系统、子系统或类的行为进行建模。每张图都显示一组用况、参与者以及它们之间的关系。【在第19章中讨论活动图；在第24章中讨论状态图；在第18章中讨论顺序图和协作图。】

用况图用于对一个系统的用况视图建模。多数情况下包括对系统、子系统或类的语境建模，或者对这些元素的行为需求建模。

用况图对可视化、详述和文档化一个元素的行为是非常重要的，它们通过表示元素如何在语境中被使用的外部视图，使系统、子系统和类易于探讨和理解。另外，用况图对通过正向工程来测试可执行的系统和通过逆向工程来理解可执行的系统也是很重要的。

17.1　入门

假设别人递给你一个盒子，在盒子的一面上有一些按钮和一个小的LCD板，除了这些，盒

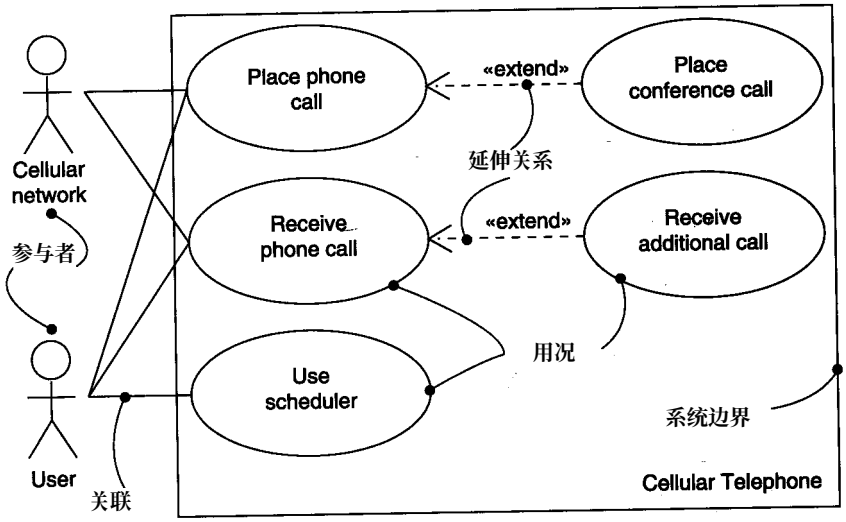

图17-1　用况图

子上没有任何说明信息，你甚至得不到一点关于如何使用它的暗示。那么你只能随机地按动按钮，并观察会发生什么。如果不经过许多次的尝试和实验，很难立即说出这盒子是干什么的或如何正确使用它。

软件密集型系统也是这样。如果你是一名用户，那么可能会交给你一个系统并告诉你去使用它。如果这个系统遵循你所熟悉的操作系统平台的一般使用惯例，那你勉强可以开始用它来做一些有用的事。但你不可能以这种方式理解它的更复杂、更精细的行为。类似地，如果你是一个开发者，可能会交给你一个遗留系统或一组构件并告诉你去使用它。在你对它们的用法形成一个概念模型之前，你很难立即知道如何使用那些元素。

在UML中，用况图用于对系统、子系统或类的行为进行可视化，以使用户能够理解如何使用这些元素，并使开发者能够实现这些元素。如图 17-1所示，你可提供一个用况图来对那个盒子（即大多数人都会拨打蜂窝网移动电话手机）的行为进行建模。

17.2 术语和概念

用况图（*use case diagram*）是显示一组用况、参与者以及它们之间关系的图。

1. 公共特性

用况图只是图的一种特殊类型，它具有与所有其他图一样的公共特征，即一个名称以及投影到模型上的图形化的内容。用况图与其他各种图不同的是其特殊的内容。【在第7章中讨论图的一般特性。】

2. 内容

用况图通常包括：

- 用况
- 参与者
- 依赖、泛化以及关联关系

与所有其他图一样，用况图还包含注解和约束。

用况图还可以含有包，用来将模型中的元素组合成更大的组块。偶尔，尤其是要可视化一个特殊的执行系统时，还可以把用况的实例引入到图中。【在第16章中讨论用况和参与者；在第 5章和第10章中讨论关系；在第12章中讨论包；在第13章中讨论实例。】

3. 一般应用

用况图用于对系统的静态用况视图进行建模。这个视图主要支持系统的行为，即该系统在它的周边环境的语境中所提供的外部可见服务。【在第2章中讨论用况视图。】

当你对系统的静态用况视图建模时，通常，你会用以下两种方式之一来使用用况图：

1）对系统的语境建模

对一个系统的语境进行建模，包括围绕整个系统画一条线，并声明有哪些参与者位于系统之外并与系统进行交互。在这里，用况图说明了参与者以及他们所扮演的角色的含义。

2）对系统的需求建模

对一个系统的需求进行建模，包括说明这个系统应该做什么（从系统外部的一个视点出发），

而不考虑系统应该怎样做。在这里，用况图说明了系统想要的行为。通过这种方式，用况图使我们能够把整个系统看作一个黑盒子；你可以观察到系统外部有什么，系统怎样与那些外部事物相互作用，但却看不到系统内部是如何工作的。【在第4章和第6章中讨论需求。】

17.3 普通建模技术

17.3.1 对系统的语境建模

给定一个系统——任意系统，会有一些事物存在于系统的内部，一些事物存在于该系统的外部。例如，在一个信用卡验证系统中，账户、事务处理和欺诈行为检测代理均存在于系统内部，而像信用卡顾客和零售机构这样的事物则存在于系统的外部。存在于系统内部的事物的职责是完成系统外部事物期望系统提供的行为。所有存在于系统外部并与系统进行交互的事物构成了该系统的语境。语境定义了系统存在的环境。

在UML 中，用用况图对系统的语境进行建模，所强调的是围绕在系统周围的参与者。决定什么作为参与者是重要的，因为这样做说明了与系统进行交互的一类事物。决定什么不作为参与者也同样重要，甚至更为重要，因为它限定了系统的环境，使之只包含那些在系统的生命周期中所必需的参与者。【在第31章中讨论系统。】

对系统的语境建模，要遵循如下策略：

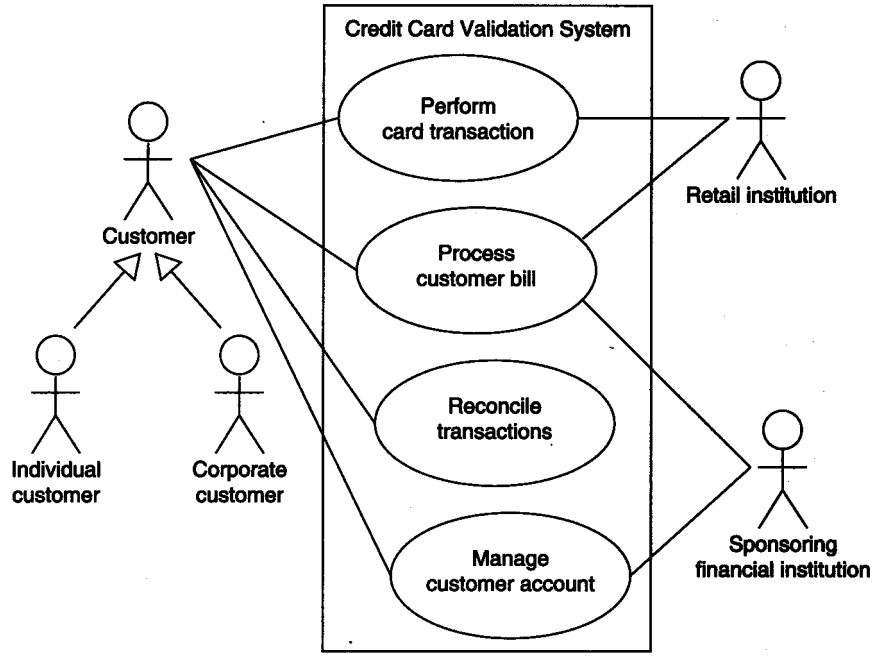

图17-2 对系统的语境建模

- 用以下几组事物来识别系统周围的参与者：需要从系统中得到帮助以完成其任务的组；执行系统的功能时所必需的组；与外部硬件或其他软件系统进行交互的组；以及为了管理和维护而执行某些辅助功能的组。
- 将相类似的参与者组织成一般/特殊的结构层次。
- 在需要加深理解的地方，为每个参与者提供一个构造型。
- 将这些参与者放入用况图中，并说明从每个参与者到系统的用况之间的通信路径。

例如，图17-2显示了一个信用卡验证系统的语境，它强调围绕在系统周围的参与者。其中有顾客（Customer），分为两类（单独顾客（Individual customer）和合作顾客（Corporate customer））。这些参与者是人与系统交互时所扮演的角色。在这个语境中，还有代表机构的参与者，如零售机构（Retail institution）（顾客通过这个机构刷卡，购买商品或服务）、主办财务机构（Sponsoring financial institution）（负责信用卡账户的结帐服务）。在现实世界中，后两个参与者本身就可能是一个软件密集型系统。

同样的技术也用于对子系统的语境建模。处于某一抽象层次上的一个系统常常是处于更高抽象层次上的一个更大的系统的子系统。因此，在你建造由若干相互连接的系统构成的大系统时，对子系统的语境建模是非常有用的。【在第31章中讨论子系统。】

17.3.2 对系统的需求建模

需求是一个系统的设计特性、特征或行为。当陈述一个系统的需求时，就相当于陈述建立在系统外部的事物与系统之间的一份契约，契约上声明了期望系统做什么事。在大多数情况下，你并不关心系统怎么做，只关心它做什么。一个行为良好的系统能够可信地、可预料地和可靠地完成它所有的需求。当建造一个系统时，重要的是以关于系统应该做什么的协定为起始点，尽管当你迭代地、增量地实现系统时，你对那些需求的理解肯定还会演化。类似地，当你把一个系统交付使用时，知道它将怎样行动是正确使用它的关键。

可以用各种形式表达需求，从非结构化的文字描述到形式语言的表达，以及介于两者之间的其他任意形式。大多数的系统的功能需求都可以以用况来表达， UML的用况图对管理这些需求是不可缺少的。【节点可用于描述状态需求，这在第6章中讨论。】

对系统的需求建模，要遵循如下策略：

- 通过识别系统周围的参与者来建立系统的语境。
- 对于每个参与者，考虑它期望的行为或需要系统提供的行为。
- 把这些公共的行为命名为用况。
- 分解公共行为，放入到新的用况中以供其他的用况使用；分解异常行为，放入新的用况中以延伸较为主要的控制流。
- 在用况图中对这些用况、参与者以及它们的关系进行建模。
- 用陈述非功能需求的注解修饰这些用况；可能还要把其中的一些附加到整个系统。

图17-3是上一个用况图的扩充。尽管没有画出参与者与用况之间的关系，但加入了额外的用况，这些用况对于一般的顾客不可见，但仍是系统的基本行为。这张图是有价值的，因为它为最终用户、领域专家以及开发者提供了一个一致的最初的交流场所，以便可视化、详述、构造

和文档化他们关于系统的功能需求的决策。例如，检测卡的真伪（Detect card fraud）对于零售机构（Retail institution）和主办财务机构（Sponsoring financial institution）都是很重要的行为。类似地，报告账户的状态（Report on account status）是系统的语境中不同机构所需要的另一个行为。

由用况Manage Network outage所建模的需求，与所有其他的用况有一点不同，因为它表示的是为保证系统的可靠性和不间断操作所需的辅助行为。【在第23章中讨论对负载均衡和网络配置的动态建模。】

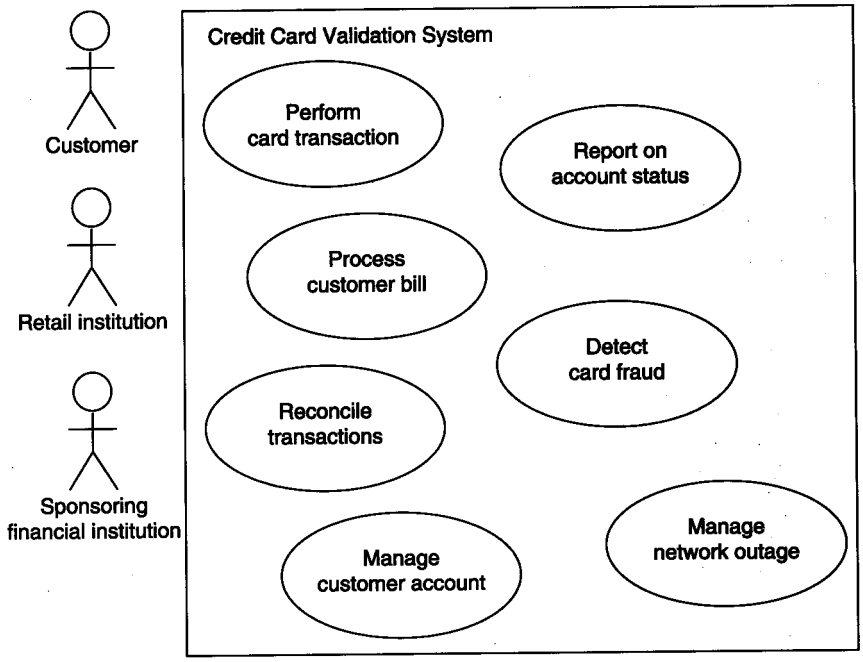

图17-3 对系统的需求建模

这一技术同样可以应用于对子系统的需求建模。【在第31章中讨论子系统。】

17.3.3 正向工程和逆向工程

UML中的大部分其他图，包括类图、构件图和状态图，都可以在正向工程和逆向工程中选用，因为它们都在可执行的系统中有一个类似物。用况图则有一点不同，它反映但并不详述一个系统、子系统或类的实现过程。用况描述元素怎样行动，而不描述该行为如何被实现，所以，不能直接地对它进行正向或逆向工程。【在第7章中讨论图；在第16章中讨论用况。】

正向工程（*forward engineering*）是通过到一个实现语言的映射，把一个模型转换为代码的过程。用况图可通过正向工程，形成对它所应用的元素的测试。用况图中的每个用况说明一个事件流（或这些流的变体），这些流说明了元素被期望如何行动——这正是值得测试的。一个结构良好的用况甚至说明了前置和后置条件，可用来定义一个测试的初态和它的成功判定标准。对于用况图中的每个用况，你都可以创建一个测试用例，当每次发布这个元素的一个新版本时

都可以运行该用例，从而在其他元素依赖它之前就保证它能像要求的那样工作。

对用况图进行正向工程，应遵循如下策略：

- 对于图中的每个用况，辨别它的事件流和异常事件流。
- 根据你选择的测试深度，为每个流产生一个测试底稿，把流的前置条件作为测试的初态，把流的后置条件作为流的成功判定标准。
- 根据需要，产生一个测试支架来描述每个与用况交互的参与者。把信息传给元素的参与者或是通过元素来执行的参与者都可以被现实世界的等价物模拟或替换。
- 每次发布用况图应用的元素时，都使用工具来运行相应的测试。

逆向工程（*reverse engineering*）是通过把特定的实现语言映射成一幅图，来把一个代码转换为模型的过程。自动地对用况图进行逆向工程，大大地超出了当前的技艺范畴，原因是从说明一个元素如何动作到说明它如何实现，会丢失一些信息。但是，你可以研究一个已存在的系统，并手工地辨别出它想要实现的行为，然后生成一张用况图。事实上，每次当你接收到一个没有文档的软件体时，你正是这样做的。 UML的用况图只是简单地向你提供一种标准的、表现力强的语言，来陈述你的发现。

对用况图进行逆向工程，要遵循如下策略：

- 识别与系统进行交互的每一个参与者。
- 对于每个参与者，考虑它与系统交互的方式，改变该系统的状态或环境的方式，或响应某些事件的方式。
- 跟踪可执行系统中与每个参与者有关的事件流。从主要的流开始，并随后考虑可选择的路径。
- 通过声明一个对应的用况，将相关的流簇集在一起。考虑用延伸关系对变体建模，用包含关系对公共的流建模。
- 将这些参与者和用况放入一张用况图中，并建立它们之间的关系。

17.4 提示和技巧

在UML中创建用况图时，应当记住每个用况图只是一个系统的静态用况视图的图形化表示。也就是说，一个单独的用况图不必捕捉一个系统的用况视图的所有事情。将一个系统的所有用况图收集在一起，就表示该系统的完整的静态用况视图；每一个用况图只单独地表示一个方面。

一个结构良好的用况图，应满足如下的要求：

- 关注于与一个系统的静态用况视图的一个方面的通信。
- 只包含那些对于理解这方面必不可少的用况和参与者。
- 提供与它的抽象层次相一致的详细表示；只能加入那些对于理解问题必不可少的修饰（如延伸点）。
- 不应该过分简化和抽象信息，以致使读者误解重要的语义。

当绘制一张用况图时，要遵循如下的策略：

- 给出一个表达其目的的名称。

- 摆放元素时，尽量减少线的交叉。
- 从空间上组织元素，使得在语义上接近的行为和角色在物理位置上也接近。
- 使用注解和颜色作为可视化提示，以突出图的重要的特征。
- 尝试不显示太多的关系种类。一般来说，如果含有复杂的包含和延伸关系，要将这些元素放入另一张图中。