第 10 章

Software Engineering: A Practitioner's Approach, Eighth Edition

需求建模：基于类的方法

要点浏览

概念： 软件问题总是以一套交互对象为特征，借此在系统内表达每一个感兴趣的事物。每个对象变成对象类中的一个成员。对象的状态描述了每个对象，即数据属性描述了对象。我们可以用基于类的需求建模方法表达上述所有内容。

人员： 软件工程师（有时被称作分析师）使用从客户那里获取的需求来建立基于类的模型。

重要性： 基于类的需求模型利用客户视角下应用或系统中的对象。这个模型描述了常规客户的系统视图。因此，客户能恰当地进行评估，并尽早获得有用的反馈信息。然后，在重新定义模型时，它将成为软件设计的基础。

步骤： 基于类的建模定义了对象、属性和关系。对一个问题描述做一番简单的探究后，能从问题的陈述中开发外部对象和类，并以基于文本或图表的形式进行表达。在创建了模型的雏形以后，就要对其不断改进，分析和评估其清晰性、完整性和一致性。

工作产品： 可以为需求建模选择大量的基于文本和图表形式的分析模型，每种表达方法都提供了一个或多个模型元素的视图。

质量保证措施： 必须评审需求建模工作产品的正确性、完整性和一致性。必须反映所有利益相关者的要求并建立一个可以从中导出设计的基础。

关键概念

分析类
分析包
关联
属性
协作性
CRC 建模
依赖性
语法解析
运维
职责

20 世纪 90 年代早期，第一次引入基于类的需求建模方法时，常以面向对象分析作为分类方法。虽然有大量不同的基于类的方法和表达方式，但 Coad 和 Yourdon[Coa91] 为所有人注明了其统一特征：

> 面向对象方法基于的都是我们最初在幼儿园中学到的内容：对象和属性，全局和部分，类和成员。

在需求建模中使用基于类的方法可以精巧地表达这些常规内容，使非技术型的利益相关者理解项目。随着需求模型的细化和扩展，它还将包含一份规格说明，以帮助软件工程师创建软件的设计部分。

基于类建模表示了系统操作的对象、应用于对象间能有效控制的操作（也称为方法或服务）、这些对象间（某种层级）的关系以及已定义类之间的协作。基于类的分析模型的元素包括类和对象、属性、操作、CRC 模型、协作图和包。下面几节将提供一系列有助于识别和表示这些元素的非正式指导原则。

10.1 识别分析类

当你环顾房间时，就可以发现一组容易识别、分类和定义（就属性和操作而言）的物理

对象。但当你“环顾”软件应用的问题空间时，了解类（和对象）就没有那么容易了。

引述 真正困难的问题是首先发现什么才是正确的对象（类）。
Carl Argila

通过检查需求模型（第9章）开发的使用场景，并对系统开发的用例进行“语法解析”[Abb83]，我们就可以开始进行类的识别了。带有下划线的每个名词或名词词组可以确定为类，将这些名词输入到一个简单的表中，并标注出同义词。如果要求某个类（名词）实现一个解决方案，那么这个类就是解决方案空间的一部分；否则，如果只要求某个类描述一个解决方案，那么这个类就是问题空间的一部分。

提问 分析类如何把自己表现为解决方案空间的元素？

一旦分离出所有的名词，我们该寻找什么？分析类表现为如下方式之一。

- 外部实体（例如其他系统、设备、人员）：产生或使用基于计算机系统的信息。
- 事物（例如报告、显示、字母、信号）：问题信息域的一部分。
- 偶发事件或事件（例如所有权转移或完成机器人的一组移动动作）：在系统操作环境内发生。
- 角色（例如经理、工程师、销售人员）：由和系统交互的人员扮演。
- 组织单元（例如部门、组、团队）：和某个应用系统相关。
- 场地（例如制造车间或码头）：建立问题的环境和系统的整体功能。
- 结构（例如传感器、四轮交通工具、计算机）：定义了对象的类或与对象相关的类。

185 这种分类只是文献中已提出的大量分类之一[⊖]。例如，Budd[Bud96]建议了一种类的分类法，包括数据产生者（源点）、数据使用者（汇点）、数据管理者、查看或观察者类以及帮助类。

还需要特别注意的是：什么不能是类或对象。通常，决不应该使用“命令过程式的名称”为类命名[Cas89]。例如，如果医疗图像系统的软件开发人员使用名字“InvertImage”甚至“ImageInversion”定义对象，就可能犯下一个小小的错误。从软件获得的Image当然可能是一个类（这是信息域中的一部分），图像的翻转是适用于该对象的一个操作，很可能将翻转定义为对于对象Image的一个操作，但是不可能定义单独的类来暗示“图像翻转”。如Cashman[Cas89]所言：“面向对象的目的是封装，但仍保持独立的数据以及对数据的操作。”

为了说明在建模的早期阶段如何定义分析类，考虑对SafeHome安全功能的“处理说明”[⊖]进行语法解析（对第一次出现的名词加下划线，第一次出现的动词采用斜体）。

建议 虽然语法解析不能保证万无一失，但如果你正在定义数据对象及其操作的转变，语法解析会让你飞跃上一个非常出色的起始点。

SafeHome *安全功能允许房主在安装时配置安全系统，监控所有连接到安全系统的传感器，通过互联网、计算机或控制面板和房主交互信息。*

在安装过程中，用SafeHome个人计算机来设计和配置系统。为每个传感器分配编号和类型，用主密码控制启动和关闭系统，而且当传感器事件发生时会拨打输入的电话号码。

识别出一个传感器事件时，软件激活装在系统上的发声警报，由房主在系统配置活动中指定的延迟时间结束后，软件拨打监控服务的电话号码并提供位置信息，报告检测到的事件性质。电话号码将每隔20秒重拨一次，直至电话接通。

⊖ 另一个重要的分类是指定义实体、边界和控制类，在10.5节中讨论。

⊖ “处理说明”类似于用例的风格，但目标稍有不同。“处理说明”提供了将要开发的功能的整体说明，而不是从某个参与者的角度写的场景。但是要注意很重要的一点，在需求收集（获取）部分也会使用语法解析来开发每个用例。

房主通过控制面板、个人计算机或浏览器（统称为接口）来接收安全信息。接口在控制面板、计算机或浏览器窗口中显示提示信息和系统状态信息。房主采用如下形式进行交互活动……

186

抽取这些名词，可以获得如下表所示的一些潜在类：

潜在类	一般分类	潜在类	一般分类
房主	角色或外部实体	主密码	事物
传感器	外部实体	电话号码	事物
控制面板	外部实体	传感器事件	事件
安装	事件	发声警报	外部实体
系统（别名安全系统）	事物	监控服务	组织单元或外部实体
编号，类型	不是对象，是传感器的属性		

这个表应不断完善，直到已经考虑到了处理说明中所有的名词。注意，我们称列表中的每一输入项为“潜在”对象，在进行最终决定之前还必须对每一项都深思熟虑。

提问　如何确定某个潜在类是否应该真的成为一个分析类？

Coad 和 Yourdon[Coa91] 建议了 6 个选择特征，在分析模型中，分析师考虑每个潜在类是否应该使用如下这些特征。

1. 保留信息。只有记录潜在类的信息才能保证系统正常工作，这样潜在类才能在分析过程中发挥作用。
2. 所需服务。潜在类必须具有一组可确认的操作，这组操作能用某种方式改变类的属性值。
3. 多个属性。在需求分析过程中，焦点应在于“主”信息；事实上，只有一个属性的类可能在设计中有用，但是在分析活动阶段，最好把它作为另一个类的某个属性。
4. 公共属性。可以为潜在类定义一组属性，这些属性适用于类的所有实例。
5. 公共操作。可以为潜在类定义一组操作，这些操作适用于类的所有实例。
6. 必要需求。在问题空间中出现的外部实体，以及任何系统解决方案运行时所必需的生产或消费信息，几乎都被定义为需求模型中的类。

引述　阶级斗争中，一些阶级胜利了，一些阶级被消灭了……
毛泽东

187

考虑包含在需求模型中的合法类，潜在类应全部（或几乎全部）满足这些特征。判定潜在类是否应包含在分析模型中多少有点主观，而且后面的评估可能会舍弃或恢复某个类。然而，基于类建模的首要步骤就是定义类，因此必须进行决策（即使是主观的）。以此为指导，根据上述选择特征进行了筛选，分析师列出 SafeHome 潜在类，如下表所示。

潜在类	适用的特征编号
房主	拒绝：6 适用，但是 1、2 不符合
传感器	接受：所有都适用
控制面板	接受：所有都适用
安装	拒绝
系统（别名安全系统）	接受：所有都适用
编号，类型	拒绝：3 不符合，这是传感器的属性
主密码	拒绝：3 不符合
电话号码	拒绝：3 不符合
传感器事件	接受：所有都适用
发声警报	接受：2、3、4、5、6 适用
监控服务	拒绝：6 适用，但是 1、2 不符合

应注意到：（1）上表并不全面，必须添加其他类以使模型更完整；（2）某些被拒绝的潜在类将成为被接受类的属性（例如，编号和类型是 Sensor 的属性，主密码和电话号码可能成为 System 的属性）；（3）对问题的不同陈述可能导致做出“接受或拒绝”的不同决定（例如，如果每个房主都有个人密码或通过声音确认，Homeowner 类就有可能接受并满足特征 1 和 2）。

10.2　描述属性

属性描述了已经选择包含在需求模型中的类。实质上，属性定义了类，以澄清类在问题空间的环境下意味着什么。例如，如果我们建立一个系统来跟踪职业棒球手的统计信息，那么类 Player 的属性与用于职业棒球手的养老系统中的属性就是截然不同的。前者，属性可能涉及名字、位置、平均击球次数、担任防守百分比、从业年限、比赛次数等相关信息。后者，以上某些属性仍是有意义的，但另外一些属性将会更换（或扩充），例如，平均工资、充分享受优惠权后的信用、所选的养老计划、邮件地址等。

关键点　属性是在问题的环境下完整定义类的数据对象集合。

为了给分析类开发一个有意义的属性集合，软件工程师应该研究用例并选择那些合理的“属于”类的“事物”。此外，每个类都应回答如下问题：什么数据项（组合项或基本项）能
188 够在当前问题环境内完整地定义这个类？

为了说明这个问题，考虑为 SafeHome 定义 System 类。房主可以配置安全功能以反映传感器信息、报警响应信息、激活或者关闭信息、识别信息等。我们可以用如下方式表现这些组合数据项：

识别信息 = 系统编号 + 确认电话号码 + 系统状态
报警应答信息 = 延迟时间 + 电话号码
激活或者关闭信息 = 主密码 + 允许重试次数 + 临时密码

等式右边的每一个数据项可以进一步地细化到基础级，但是考虑到我们的目标，可以为 System 类组成一个合理的属性列表（图 10-1 中的阴影部分）。

传感器是整个 SafeHome 系统的一部分，但是并没有列出如图 10-1 所示的数据项或属性。已经定义 Sensor 为类，多个 Sensor 对象将和 System 类关联。通常，如果有超过一个项和某个类相关联，就应避免把这个项定义为属性。

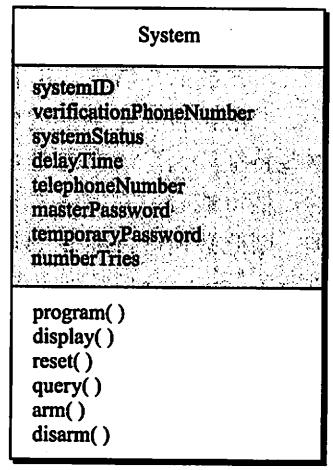


图 10-1　System 类的类图

10.3　定义操作

操作定义了某个对象的行为。尽管存在很多不同类型的操作，但通常可以粗略地划分为 4 种类型：（1）以某种方式操作数据（例如添加、删除、重新格式化、选择）；（2）执行计算的操作；（3）请求某个对象的状态的操作；（4）监视某个对象发生某个控制事件的操作。这些功能通过在属性或相关属性（10.5 节）上的
189 操作来实现。因此，操作必须“理解”类的属性和相关属性的性质。

建议　当为分析类定义操作时，集中于面向问题的行为而不是实现所要求的行为。

在第一次迭代要导出一组分析类的操作时，可以再次研究处理说明（或用例）并合理地选择属于该类的操作。为了实现这个目标，可以再次研究语法解析并分离动词。这些动词中的一部分将是合法的操作并能够很容

易地连接到某个特定类。例如，从本章前面提出的 SafeHome 处理说明中可以看到，“为传感器分配编号和类型”“主密码用于激活和解除系统”，这些短语表明：

- assign() 操作和 Sensor 类相关联。
- program() 操作应用于 System 类。
- arm() 和 disarm() 应用于 System 类。

再进一步研究，program() 操作很可能被划分为一些配置系统所需要的更具体的子操作。例如，program() 隐含着电话号码、配置系统特性（如创建传感器表、输入报警特征值）和输入密码。但是我们暂时把 program() 指定为一个单独的操作。

另外，对于语法解析，分析师能通过考虑对象间所发生的通信获得对其他操作的更为深入的了解。对象通过传递信息与另一个对象通信。在继续对操作进行说明之前，我们探测到了更详实的信息。

SafeHome 类模型

[**场景**] Ed 的小房间，开始进行分析建模。

[**人物**] Jamie、Vinod 和 Ed，SafeHome 软件工程团队的成员。

[**对话**]

Ed 已经从 ACS-DCV（本章前面的 SafeHome 中已有介绍）的用例模板中做了提取类方面的工作，并向他的同事展示了已经提取的类。

Ed：那么当房主希望选择一个摄像机的时候，他必须从一个平面设计图中进行选择。我已经定义了一个 FloorPlan 类，这个图在这里。

（他们查看图 10-2。）

Jamie：FloorPlan 这个类把墙、门、窗和摄像机都组织在一起。这就是那些标记线的意义，是吗？

Ed：是的，它们被称作“关联”，一个类根据我在图中所表示的关联关系和另一个类相关联（在 10.5 节中讨论关联）。

Vinod：那么实际的平面设计图是由墙构成的，并包含摄像机和放置在那些墙中的传感器。平面设计图如何知道在哪里放置那些对象？

Ed：平面设计图不知道，但是其他类知道。例如查看属性 WallSegment，该属性用于构建墙，墙段（WallSegment）具有起点坐标和终点坐标，其他由 draw() 操作完成。

Jamie：这些也适用于门窗。看起来摄像机有一些额外的属性。

Ed：是的，我要求它们提供转动信息和缩放信息。

Vinod：我有个问题，为什么摄像机有 ID 编号而其他对象没有呢？我注意到有个属性叫 nextWall。WallSegment 如何知道什么是下一堵墙？

Ed：好问题，但正如他们所说，那是由设计决定的，所以我将推迟这个问题直到……

Jamie：让我休息一下……我打赌你已经想出办法了。

Ed（羞怯地笑了笑）：确实，当我们开始设计时我要采用列表结构来建模。如果你坚信分析和设计是分离的，那么我安排的详细程度等级就有问题了。 190

Jamie：对我而言看上去非常好。只是我还有一些问题。

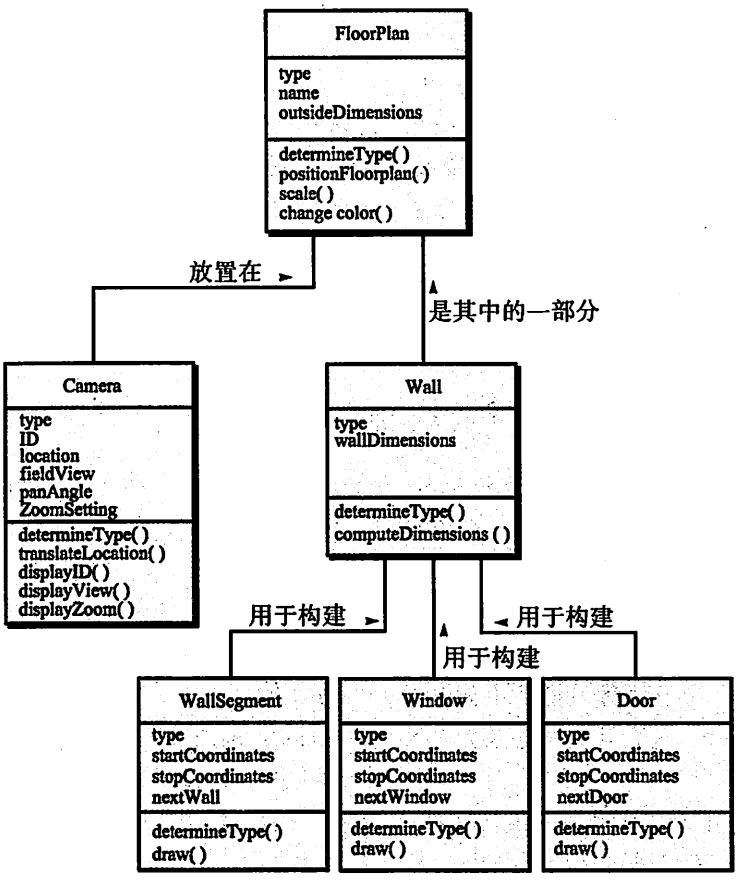
（Jamie 问了一些问题，因此做了一些小的修改。）

Vinod：你有每个对象的 CRC 卡吗？如果有，我们应该进行角色演练，以确保没有

任何遗漏。

Ed：我不太清楚如何做。

Vinod：这不难，而且确实有用，我给你演示一下。



191

图 10-2　FloorPlan 类的类图

10.4　类－职责－协作者建模

引述　使用 CRC 卡的一个目的是尽早舍弃、频繁舍弃以及低成本舍弃。事实上抽出一叠卡片比改编大量源代码要容易得多。

C. Horstmann

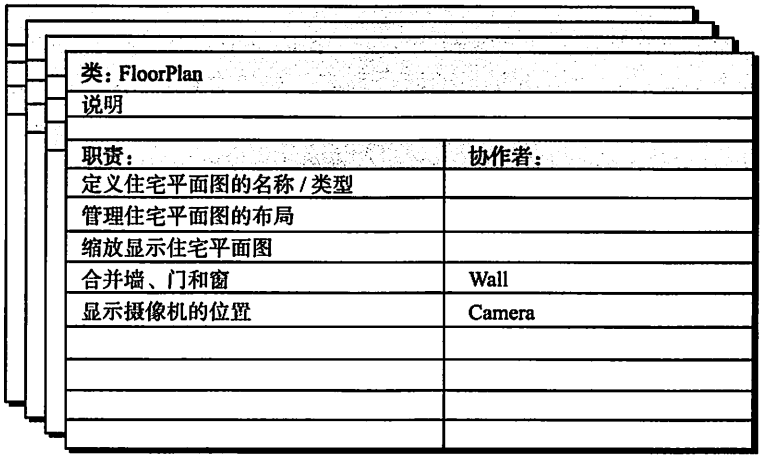
类－职责－协作者（Class-Responsibility-Collaborator，CRC）建模[Wir90]提供了一个简单方法，可以识别和组织与系统或产品需求相关的类。Ambler[Amb95]用如下文字解释了 CRC 建模：

CRC 模型实际上是表示类的标准索引卡片的集合。这些卡片分三部分，顶部写类名，卡片主体左侧部分列出类的职责，右侧部分列出类的协作者。

事实上，CRC 模型可以使用真实的或虚拟的索引卡，意图是有组织地表示类。职责是和类相关的属性和操作。简单地说，职责就是“类所知道或能做的任何事”[Amb95]。协作者是提供完成某个职责所需要信息的类。通常，协作意味着信息请求或某个动作请求。

FloorPlan 类的一个简单 CRC 索引卡如图 10-3 所示。CRC 卡上所列出的职责只是初步

的，可以添加或修改。在职责栏右边的 Wall 和 Camera 是需要协作的类。



类：FloorPlan	
说明	
职责：	协作者：
定义住宅平面图的名称 / 类型	
管理住宅平面图的布局	
缩放显示住宅平面图	
合并墙、门和窗	Wall
显示摄像机的位置	Camera

图 10-3　CRC 模型索引卡

类。在本章前面已经说明了识别类和对象的基本原则。10.1 节所说的类的分类可以通过如下分类方式进行扩展：

- *实体类*，也称作*模型或业务类*，是从问题说明中直接提取出来的（例如 FloorPlan 和 Sensor）。这些类一般代表保存在数据库中和贯穿在应用程序中（除非被明确删除）的事物。
- *边界类*用于创建用户可见的和在使用软件时交互的接口（如交互屏幕或打印的报表）。实体类包含对用户来说很重要的信息，但是并不显示这些信息。边界类的职责是管理实体对象呈现给用户的方式。例如，Camera Window 的边界类负责显示 SafeHome 系统监视摄像机的输出。
- *控制类*自始至终管理“工作单元”。也就是说，控制类可以管理：（1）实体类的创建或更新；（2）边界类从实体对象获取信息后的实例化；（3）对象集合间的复杂通信；（4）对象间或用户和应用系统间交换数据的确认。通常，直到设计开始时才开始考虑控制类。

网络资源　关于这些类的类型的精彩讨论可以参阅 http://www.oracle.com/technetwork/develop-tools/jdev/gettingstartedwithumlclassmodeling-130316.pdf。

192

引述　对象可以科学地分为三个主要类别：不工作的、损坏的和丢失的。

Russell Baker

职责。在 10.2 节和 10.3 节中已经说明了识别职责（属性和操作）的基本原则。Wirfs-Brock 和她的同事 [Wir90] 在给类分配职责时建议了以下 5 个指导原则。

提问　为类分配职责时可以采用什么指导原则？

1. **智能系统应分布在所有类中以求最大程度地满足问题的需求。**每个应用系统都包含一定程度的智能，也就是系统所知道的以及所能完成的。智能在类中可以有多种分布方式。建模时可以把“不灵巧”（Dumb）类（几乎没有职责的类）作为一些“灵巧”类（有很多职责的类）的从属。尽管该方法使得系统中的控制流简单易懂，但同时有如下缺点：把所有的智能集中在少数类，使得变更更为困难；将会需要更多的类，因此需要更多的开发工作。

   如果智能系统更平均地分布在应用系统的所有类中，每个对象只了解和执行某些

事情（通常是适度集中），并提高系统的内聚性[⊖]，这将提高软件的可维护性并减少变
193 更的副作用。

为了确定分布智能系统是否恰当，应该评估每个 CRC 模型索引卡上标记的职责，以确定某个类是否应该具有超长的职责列表。如果有这种情况就表明智能太集中[⊖]。此外，每个类的职责应表现在同一抽象层上。例如在聚合类 CheckingAccount 操作列表中，评审人员注意到两项职责：*账户余额和已结算的支票*。第一个操作的职责意味着复杂的算术和逻辑过程，第二个操作的职责是指简单的办事员活动。既然这两个操作不是相同的抽象级别，因此*已结算的支票*应该被放在 CheckEntry 的职责中，这是由聚合类 CheckingAccount 压缩得到的一个类。

2. **每个职责的说明应尽可能具有普遍性。**这条指导原则意味着应在类的层级结构的上层保持职责（属性和操作）的通用性（因为它们更有一般性，将适用于所有的子类）。
3. **信息和与之相关的行为应放在同一个类中。**这实现了面向对象原则中的封装，数据和操作数据的处理应包装在一个内聚单元中。
4. **某个事物的信息应局限于一个类中而不要分布在多个类中。**通常应由一个单独的类负责保存和操作某特定类型的信息。通常这个职责不应由多个类分担。如果信息是分布的，软件将变得更加难以维护，测试也会面临更多挑战。
5. **适合时，职责应由相关类共享。**很多情况下，各种相关对象必须在同一时间展示同样的行为。例如，考虑一个视频游戏，必须显示如下类：Player、PlayerBody、PlayerArms、PlayerLegs 和 PlayerHead。每个类都有各自的属性（例如 position、orientation、color 和 speed），并且所有属性都必须在用户操纵游戏杆时进行更新和显示。因此，每个对象必须共享职责 update 和 display。Player 知道在什么时候发生了某些变化并且需要 update 操作。它和其他对象协作获得新的位置或方向，但是每个对象
194 控制各自的显示。

协作。类用一种或两种方法来实现其职责：（1）类可以使用其自身的操作控制各自的属性，从而实现特定的职责；（2）类可以和其他类协作。Wirfs-Brock 和她的同事 [Wir90] 这样定义协作：

协作是以客户职责实现的角度表现从客户到服务器的请求。协作是客户和服务器之间契约的具体实现……如果为了实现某个职责需要发送任何消息给另一个对象，我们就说这个对象和其他对象有协作。单独的协作是单向流，即表示从客户到服务器的请求。从客户的角度看，每个协作都和服务器的某个特定职责实现相关。

要识别协作可以通过确认类本身是否能够实现自身的每个职责。如果不能实现每个职责，那么需要和其他类交互，因此就要有协作。

例如，考虑 SafeHome 的安全功能。作为活动流程的一部分，ControlPanel 对象必须确定是否启动所有的传感器，定义名为 determine-sensor-status() 的职责。如果传感器是开启的，那么 ControlPanel 必须设置属性 status 为“未准备好”。传感器信息可以从每个 Sensor 对象获取，因此只有当 ControlPanel 和 Sensor 协作时才能实现 determine-sensor-status() 职责。

⊖ 内聚性是一个设计概念，将在第 12 章中讨论。

⊖ 在这种情况下，可能需要将一个类分成多个类或子系统，以便更有效地分布智能。

为帮助识别协作者，分析师可以检查类之间三种不同的通用关系 [Wir90]：（1）is-part-of（是……一部分）关系；（2）has-knowledge-of（有……的知识）关系；（3）depends-upon（依赖……）关系。在下面的段落中将简单地分别说明这三种通用关系。

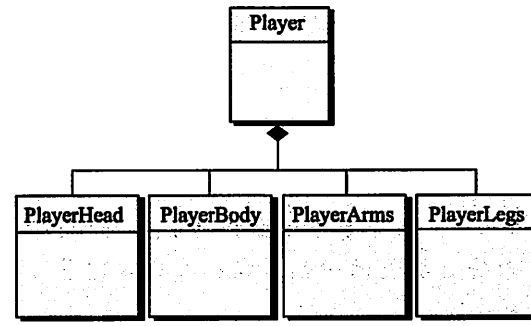


图 10-4　复合聚合类

属于某个聚合类一部分的所有类可通过 is-part-of 关系和聚合类连接。考虑前面提到的视频游戏中所定义的类，PlayerBody 是 Player 的一部分，PlayerArms、PlayerLegs 和 PlayerHead 也类似。在 UML 中，使用如图 10-4 所示的聚合方式表示这些关系。

当一个类必须从另一个类中获取信息时，就建立了 has-knowledge-of 关系。前面所说的 determine-sensor-status() 职责就是 has-knowledge-of 关系的一个例子。

depends-upon 关系意味着两个类之间具有 has-knowledge-of 和 is-part-of 不能实现的依赖关系。例如，PlayerHead 通常必须连接到 PlayerBody（除非视频游戏特别暴烈），然而每个对象并没有其他对象的直接信息。PlayerHead 对象的 center-position 属性由 PlayerBody 的中心位置确定，该信息通过第三方对象 Player 获得，即 PlayerBody 需要 Player。因此，PlayerHead 依赖 PlayerBody。 195

在所有情况下，我们都把协作类的名称记录在 CRC 模型索引卡上，紧靠在协作的职责旁边。因此，索引卡包含一个职责列表以及相关的能够实现这些职责的协作（图 10-3）。

当开发出一个完整的 CRC 模型时，利益相关者可以使用如下方法评审模型 [Amb95]。

1. 所有参加（CRC 模型）评审的人员拿到一部分 CRC 模型索引卡。拆分协作卡片（也就是说每个评审员不得有两张存在协作关系的卡片）。
2. 分类管理所有的用例场景（以及相关的用例图）。
3. 评审组长细致地阅读用例。当评审组长看到一个已命名的对象时，给拥有相应类索引卡的人员一个令牌。例如，SafeHome 的一个用例包含如下描述：

   房主观察 SafeHome 控制面板以确定系统是否已经准备接收输入。如果系统没有准备好，房主必须手工关闭窗户（门）以便指示器呈现就绪状态。（未就绪指示器意味着某个传感器是开启的，也就是说某个门或窗户是打开的。）

   当评审组长看到用例说明中的“控制面板”，就把令牌传给拥有 ControlPanel 索引卡的人员。“意味着某个传感器是开启的”语句需要索引卡包含确认该含义的职责（由 determine-sensor-status() 实现该职责）。靠近索引卡职责的是协作者 Sensor，然后令牌传给 Sensor 对象。 196
4. 当令牌传递时，该类卡的拥有者需要描述卡上记录的职责。评审组确定（一个或多个）职责是否满足用例需求。
5. 如果记录在索引卡上的职责和协作不能满足用例，就需要修改卡片。修改可能包括定义新类（和相关的 CRC 索引卡），或者在已有的卡上说明新的或修改的职责、协作。

该过程持续进行直到用例（或用例图）编写结束。评审完所有用例后将继续进行需求建模。

SafeHome CRC 模型

[场景] Ed 的办公室，刚开始需求建模。

[人物] Vinod 和 Ed，SafeHome 软件工程团队成员。

[对话]

（Vinod 已经决定通过一个例子向 Ed 展示如何开发 CRC 卡。）

Vinod：在你着手于监视功能而 Jamie 忙着安全功能的时候，我在准备住宅管理功能。

Ed：情况怎样？市场营销人员的想法总是在变化。

Vinod：这是整个功能的第一版用例……我们已经改进了一点，它应该能提供一个整体视图。

用例：SafeHome 住宅管理功能。

说明：我们希望通过个人计算机上的住宅管理接口或互联网连接，来控制有无线接口控制器的电子设备。系统应该允许我打开或关闭指定的灯，控制连接到无线接口的设备，设置取暖和空调系统达到预定温度。为此，我希望从房屋平面图上选择设备。每个设备必须在平面图上标识出来。作为可选的特性，我希望控制所有的视听设备——音响设备、电视、DVD、数字录音机等。

通过不同选项就能够针对各种情况设置整个房屋，第一个选项是“在家”，第二个是“不在家”，第三个是“彻夜不归”，第四个是“长期外出”。所有这些情况都适用于所有设备的设置。在彻夜不归和长期外出时，系统将以随机的间隔时间开灯和关灯（以造成有人在家的错觉），并控制取暖和空调系统。我应能够通过有适当密码保护的互联网撤销这些设置 .....

Ed：那些负责硬件的伙计已经设计出所有的无线接口了吗？

Vinod（微笑）：他们正在忙这个，据说没有问题。不管怎样，我从住宅管理中提取了一批类，我们可以用一个做例子。就以 HomeManagementInterface 类为例吧！

Ed：好，那么职责是什么？类及协作者们的属性和操作是那些职责所指向的类。

Vinod：我想你还不了解 CRC。

Ed：可能有点，但还是继续吧。

Vinod：这就是我给出的 HomeManagementInterface 的类定义。

属性：

optionsPanel——在按钮上提供信息，用户可以使用这些信息选择功能。

situationPanel——在按钮上提供信息，用户可以使用这些信息选择环境。

floorPlan——类似于监视对象，但这个用来显示设备。

deviceIcons——图标上的信息，代表灯、家用电器、HVAC 等。

devicePanels——模拟家用电器或设备控制面板，允许控制。

操作：

displayControl(), selectControl(), displaySituation(), selectSituation(), accessFloorplan(),selectDeviceIcon(), displayDevicePanel(), accessDevicePanel()……

类：HomeManagementInterface

职责	协作者
displayControl ( )	OptionsPanel (类)
selectControl ( )	OptionsPanel (类)
displaySituation ( )	SituationPanel (类)
selectSituation ( )	SituationPanel (类)
accessFloorplan ( )	FloorPlan (类)
.....	.....

Ed：这样，当调用 accessFloorPlan() 操作时，将和 FloorPlan 对象协作，类似我们为监视开发的对象。等一下，我这里有它的说明（他们查看图 10-2）。

Vinod：确实如此。如果我们希望评审整个类模型，可以从这个索引卡开始，然后

197

到协作者的索引卡，再到协作者的协作者的索引卡，依此类推。

Ed：这真是个发现遗漏和错误的好方法。

Vinod：的确如此。

10.5　关联和依赖

关键点　关联定义了类之间的联系，多样性定义了一个类和另一个类之间联系的数量关系。

在很多例子中，两个分析类以某种方式相互联系着。在 UML 中，这些联系被称作关联（accociation）。参考图 10-2，通过识别 FloorPlan 与另外两个类 Camera 和 Wall 之间的一组关联确定 FloorPlan 类。类 Wall 和三个构成墙的类 WallSegment、Window 和 Door 相关联。

在某些情况下，关联可以更进一步地指出多样性。参考图 10-2，一个 Wall 对象可以由一个或多个 WallSegment 对象构成。此外，Wall 对象可能包含 0 或多个 Window 对象以及 0 或多个 Door 对象。这些多样性限制如图 10-5 所示，其中“一个或多个”使用 1..* 表示，“0 或多个”使用 0..* 表示。在 UML 中，星号表示范围无上界。⊖

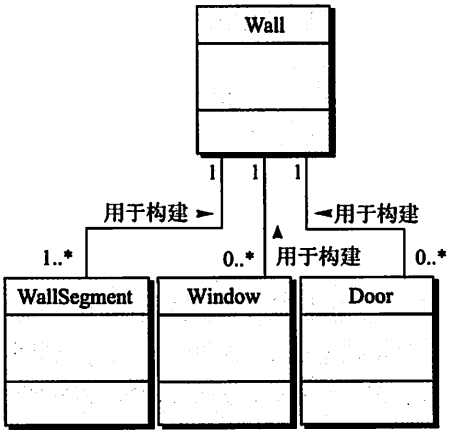


图 10-5　多样性

在很多事例中，两个分析类之间存在客户 – 服务器关系。这种情况下，客户类以某种方式依赖于服务器类并且建立了依赖关系。依赖性是由一个构造型（stereotype）定义的。在 UML 中，构造型是一个“可扩展机制”[Arl02]，允许软件工程师定义特殊的建模元素，这些建模元素的语义是由用户自定义的。在 UML 中，构造型由一对尖括号表示（如 <<stereotype>>）。

提问　什么是构造型？

198

下面举例说明 SafeHome 监视系统中的简单依赖关系。Camera 对象（本例中的服务器类）向 DisplayWindow 对象（本例中的客户类）提供视频图像。这两个对象之间的关系不是简单的关联，而是存在着依赖关系。在监视用例中（没有列出来），建模者知道必须提供特定的密码才能查看指定摄像机的位置。其实现的一种方法是让 Camera 请求密码，然后在获得 DisplayWindow 的允许后显示视频。这可以由图 10-6 表示，其中 <<access>> 意味着通过特定的密码控制对摄像机输出的使用。

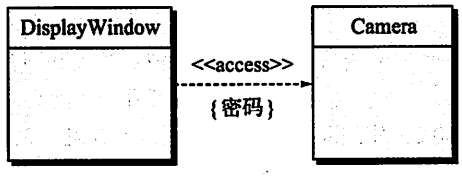


图 10-6　依赖性

10.6　分析包

关键点　分析包用来集合一组相关的类。

分析建模的一部分重要工作是分类，也就是将分析模型的各种元素（如用例、分析类）以一种方式分类，分组打包后称为分析包，并取一个有代表性的名字。

⊖ 其他的多样性关联（一对一、一对多、多对多、一对某指定上下限的范围，以及其他）可以标识为关联的一部分。

为了说明分析包的使用，考虑我们前面所说的视频游戏。假设视频游戏的分析模型已经
199 建成，并且已经生成大量的类。有一些类关注游戏环境，即用户游戏时所看到的可视场景。Tree、Landscape、Road、Wall、Bridge、Building、VisualEffect 类等可能就属于游戏环境类。另一些类关注游戏内的人物，说明他们的肢体特点、动作和局限。也可能定义了（前面提到的）Player、Protagonist、Antagonist、SupportingRoles 类。还需要一些类用来说明游戏的规则，即游戏者如何穿越环境。例如这里可以选 RulesOfMovement 类和 ConstraintsOnAction 类。还可能存在很多其他类，可以用图 10-7 中的分析包表示这些类。

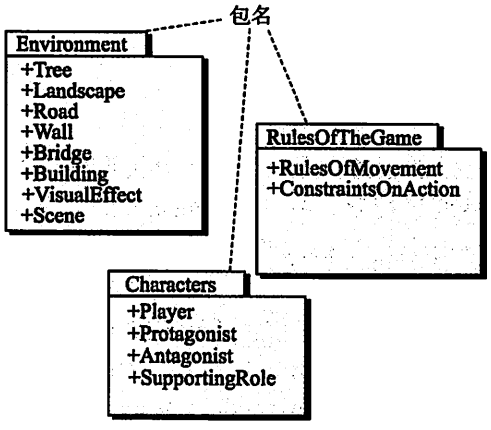


图 10-7　包

每个包中分析类名字前的加号表示该类是公共可见的，因此可以从其他包访问。尽管目前的图中没有显示，但包中的任何元素之前都可以添加其他符号。负号表示该元素对其他包是隐藏的，# 号表示只能由指定包中的类访问该元素。

10.7　小结

为了识别分析类，基于类的建模使用从用例和其他编写的应用描述中导出的信息。可以使用语法解析从文本说明中提取候选类、属性和操作，并制定用于定义类的标准。

CRC 索引卡可以用于定义类之间的联系。此外，可以使用各种 UML 建模方法定义类之
200 间的层次、关系、关联、聚合和依赖。使用分析包方式进行分类和分组，在某种意义上为大型系统提供了更好的管理。

习题与思考题

10.1　构建如下系统中的一个：

a. 你所在大学中基于网络的课程注册系统。

b. 一个计算机商店的基于 Web 的订单处理系统。

c. 一个小企业的简单发票系统。

d. 内置于电磁灶或微波炉的互联网食谱。

选择你感兴趣的系统开发一套处理说明。然后使用语法解析技术识别候选对象和类。

10.2　使用问题 10.1 中识别的类开发一系列操作。

10.3　为问题 9.5 中表述的 PHTRS 系统开发一个类模型。

10.4　为 10.4 节中非正式描述的 SafeHome 住宅管理系统编写一个基于模板的用例。

10.5　为问题 10.1 所选的产品或系统开发一组完整的 CRC 模型索引卡。

10.6　指导你的同事一起评审 CRC 索引卡，评审结果中增加了多少类、职责和协作者？

10.7　什么是分析包？如何使用分析包？

扩展阅读与信息资源

Weisfeld 讨论了常规的基于类的观点（《The Object-Oriented Thought Process》，4th ed.，Addison-Wesley，2013）。讨论基本类的建模方法的书包括：Bennet 和 Farmer（《Object-Oriented Systems

Analysis and Design Using UML》, McGraw-Hill, 2010), Ashrafi(《Object-Oriented Systems Analysis and Design》, Prentice Hall, 2008), Booch(《Object-Oriented Analysis and Design with Applications》, 3rd ed., Addison-Wesley, 2007), George 和他的同事(《Object-Oriented Systems Analysis and Design》, 2nd ed., Prentice Hall, 2006), O'Docherty(《Object-Oriented Analysis and Design》, Wiley, 2005), Satzinger 等人(《Object-Oriented Analysis and Design with the Unified Process》, Course Techonlogy, 2004), Stumpf 和 Teague(《Object-Oriented Systems Analysis and Design with UML》, Prentice Hall, 2004)。

UML 建模技术可以应用于分析和设计，讨论这方面的书包括：Dennis 和他的同事(《Systems Analysis and Design with UML Version 2.0》, Wiley, 4th ed., 2012), Ramnath 和 Dathan(《Object-Oriented Analysis and Design》, Springer, 2011), Bennett 和 Farmer(《Object-Oriented System Analysis and Design Using UML》, McGraw-Hill, 4th ed., 2010), Larman(《Applying UML and Patterns：An Introduction to Object-Oriented Analysis and Design and Iterative Development》, Dohrling Kindersley, 2008), Rosenberg 和 Stephens(《Use Case Driven Object Modeling with UML Theory and Practice》, Apress, 2007), 还有 Arlow 和 Neustadt(《UML 2 and the Unified Process》, 2nd ed., Addison-Wesley, 2005)。这些书都描述了用 UML 内容分析类的定义。

网上有很多关于基于类的方法进行需求建模的信息。可以参考 SEPA 网站 www.mhhe.com/pressman 上有关分析建模的最新参考文献。