

需求建模：场景、信息与类分析

要点浏览

概念：文字记录是极好的交流工具，但并不必然是表达计算机软件需求的最好方式。分析建模使用文字和图表的综合形式，以相对容易理解的方式描绘需求，更重要的是，可以更直接地评审它们的正确性、完整性和一致性。

人员：软件工程师（有时被称作分析师）使用从客户那里提取的需求构建模型。

重要性：为了确认软件需求，你需要从不同的视角检验需求。本章将从三个不同的视角考虑需求建模：基于场景的模型、数据（信息）模型和面向类的模型。分析建模从多个“维度”表现需求，这样就增加了查明错误、消除不一致性、发现遗漏的几率。

步骤：基于场景的建模从用户的角度表现系统；数据建模提出了信息空间同时描述了软件要加工的数据对象以及数据对象间的关系；基于类的建模定义了对象、属性和关系；行为建模描述了系统状态、类和事件在这些类上的影响。在创建了模型的雏形以后，就要对其不断改进，并分析评估其清晰性、完整性和一致性。在第7章扩展了模型维度，增加了表示方法，提供了更完善的需求视图。

工作产品：可以选择大量的图表格式用于分析模型，每种表现方法都提供了一个或多个系统元素的视图。

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导出设计的基础。

关键概念

活动图

分析类

分析包

关联

基于类建模

CRC建模

数据建模

域分析

语法解析

需求建模

基于场景建模

泳道图

UML模型

用例

在技术层面上，软件工程开始于一系列的建模工作，最终生成待开发软件的需求规格说明和设计表示。需求模型实际上是一组模型^①，是系统的第一个技术表示。

在一本关于分析建模方法的开创性书籍中Tom DeMarco[Dem79]这样描述该过程：

回顾已确认分析阶段的问题和过失，我建议对分析阶段的目标进行以下的增补。分析的结果必须是高度可维护的，尤其是要将此结果应用于目标文档[软件需求规格说明]。必须使用一种有效的分割方法解决规模问题，维多利亚时代小说式的规格说明是不行的。尽可能使用图形符号。考虑问题必须区分逻辑的[本质]和物理的[实现]……无论如何，我们至少需要……某种帮助我们划分需求的方法，并在规格说明前用文档记录该划分……某种跟踪和评估接口的手段……使用比叙述性文本更好的新工具来描述逻辑和策略。

尽管DeMarco在25年前就写下了关于分析建模的特点，但他的意见仍然适用于现代的分析建模方法和表示方法。

① 在本书过去的版本中，我使用分析模型而不是需求模型。在这一版本中我决定使用这两个用语，以便表达在解决问题的多个方面时定义的建模活动。当导出需求时分析其中的动作。

6.1 需求分析

需求分析产生软件工作特征的规格说明,指明软件和其他系统元素的接口,规定软件必须满足的约束。需求分析让软件工程师(有时这个角色也被称作分析师或建模师)细化在前期需求工程的起始、导出、谈判任务中建立的基础需求(第5章)。

“任何一个需求‘视图’都不足以理解和描述一个复杂系统所需的行为。”——
Alan M. Davis

KEY POINT

一旦软件完成后,分析模型和需求规格说明书将成为提供评估软件质量的手段。

需求不是架构。需求既不是设计,也不是用户接口。需求就是指需要的。——
Andrew Hunt 和 David Thomas

KEY POINT

分析模型应该描述什么是客户所需,应该建立设计的基础,建立有效的目标。

需求建模动作产生为以下一种或多种模型类型:

- 场景模型:出自各种系统“参与者”观点的需求。
- 数据模型:描述问题信息域模型。
- 面向类的模型:表示面向对象类(属性和操作)的模型,其方式为通过类的协作获得系统需求。
- 面向流程的模型:表示系统的功能元素并且描述当功能元素在系统中运行时怎样进行数据变换的模型。
- 行为模型:描述如何将软件行为看做是外部“事件”后续模型。

这些模型为软件设计者提供信息,这些信息可以转化为结构、接口和构件级设计。最终,在软件开发完成后,需求模型(和需求规格说明)就为开发人员和客户提供了评估软件质量的手段。

本章我关注基于场景的建模,这项技术在整个软件工程界迅猛发展;数据建模是较为特殊的技术,适用于当一个应用问题必须生成或处理一个复杂的信息空间时;面向类建模表示面向对象类和允许的系统功能间的有效协作。面向流程的模型、行为模型、基于模式的建模和Web应用模型将在第7章讨论。

6.1.1 总体目标和原理

在整个需求建模过程中,软件工程师的主要关注点集中在“做什么”而不是“怎么做”方面。包括:在特定环境下发生哪些用户交互?系统处理什么对象?系统必须执行什么功能?系统展示什么行为?定义什么接口?有什么约束^①?

在前面的章节中,我们注意到在该阶段要得到完整的需求规格说明是不可能的。客户也许无法精确地确定想要什么,开发人员也许无法确定能恰当地实现功能和性能的特定方法,这些现实都削弱了迭代需求分析和建模方法的效果。分析师将为已经知道的内容建模,并使用该模型作为软件进一步扩展的设计基础^②。

需求模型必须实现三个主要目标:(1)描述客户需要什么;(2)为软件设计奠定基础;(3)定义在软件完成后可以被确认的一组需求。分析模型在系统级描述和软件设计(第8章至第13章)之间建立了桥梁。这里的系统级描述给出了在软件、硬件、数据、人员和其他系统元素共同作用下的整个系统或商业功能,而软件设计给出了软件的应用程序结构、用户接口以及构件级的结构。这个关系如图6-1所示。

① 应该注意当客户是更为老练的技术人员时,规格说明书应侧重How同What一样重要。但是,基本观点应保留在What上。

② 或者软件团队可以花些功夫选择一个原型(第2章),以便更好地理解系统的需求。

重要的是要注意需求模型的所有元素都可以直接跟踪到设计模型。通常难以清楚地区分这两个重要的建模活动之间的设计和分析工作，有些设计总是作为分析的一部分进行，而有些分析将在设计中进行。

6.1.2 分析的经验原则

我们做需求分析时有没有可以帮助我们指南？

“值得进攻的问题总是通过反击证明其价值。”——Piet Hein

Ariow和Neustadt[Arl02]提出了大量有价值的经验原则，在创建分析模型时应遵循这些经验原则：

- 模型应关注在问题域或业务域内可见的需求，抽象的级别应该相对高一些。“不要陷入细节”[Arl02]，即不要试图解释系统将如何工作。
- 需求模型的每个元素都应能增加对软件需求的整体理解，并提供对信息域、功能和系统行为的深入理解。
- 关于基础结构和其他非功能的模型应推延到设计阶段再考虑。例如，可能需要一个数据库，但是只有在已经完成问题域分析之后才应考虑实现数据库所必需的类、访问数据库所需的功能，以及使用时所表现出的行为。
- 最小化整个系统内的关联。表现类和功能之间的联系非常重要，但是，如果“互联”的层次非常高，应该想办法减少互联。
- 确认需求模型为所有利益相关者都带来价值。对模型来说，每个客户都有自己的使用目的。例如，利益相关的业务人员将使用模型确认需求；设计人员将使用模型作为设计的基础；质量保证人员将使用模型帮助规划验收测试。
- 尽可能保持模型简洁。如果没有提供新的信息，不要添加附加图表；如果一个简单列表就够用，就不要使用复杂的表示方法。

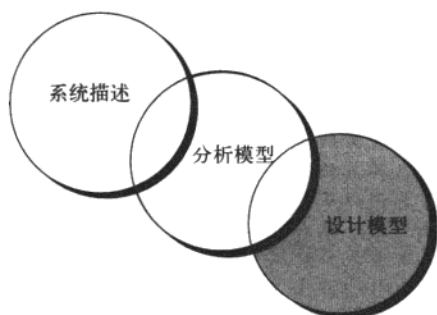


图6-1 分析模型在系统描述和设计模型之间建立桥梁

6.1.3 域分析

在需求工程讨论中（第5章），我们注意到分析模式通常在特定业务领域内的很多应用系统中重复发生。如果用一种方式对这些模式加以定义和分类，让软件工程师或分析师识别并复用这些模式，将促进分析模型的创建。更重要的是，应用可复用的设计模式和可执行的软件组件的可能性将显著增加。这将把产品投放市场的时间提前，并减少开发费用。

但问题是，首先如何识别分析模式？由谁来对分析模式进行定义和分类，并为随后的项目准备好分析模式？这些问题的答案在域分析中。Firesmith[Fir93]这样描述域分析：

软件域分析是识别、分析和详细说明某个特定应用领域的公共需求，特别是那些在该应用领域内被多个项目重复使用的需求……[面向对象的域分析是]在某个特定应用领域内，根据通用的对象、类、部件和框架，识别、分析和详细说明公共的、可复用的能力。

“特定应用领域”的范围从航空电子设备到银行业，从多媒体视频游戏到医疗设备中的嵌入式软件。域分析的目标很简单，就是：查找或创建那些广泛应用的分析类和（或）分析模式，使其能够复用^①。

① 一个域分析的补充观点是：“包括为域建模，因此软件工程师和其他的利益相关者可以更好的学习……不是所有域的类都必然导致可复用的类”[Let03]。

WebRef

可以从以下地址找到很多关于域分析的有用信息：www.iturils.com/english/SoftwareEngineering/SE_mod5.asp。

KEY POINT

域分析不关注特定的应用系统，相反关注应用所属的领域。其目的在于识别那些解决可用于域内所有应用系统的共同问题。

使用在本书前面介绍的术语，域分析可以被看做是软件过程的一个普适性活动。意思是域分析是正在进行的软件工程活动，而不是与任何一个软件项目相关的。域分析师的角色有些类似于重型机械制造业中一名优秀的刀具工的角色。刀具工的工作是设计并制造工具，这些工具可被很多人用来进行类似的而不一定是同样的工作。域分析师^①的角色是发现和定义可复用的分析模式、分析类和相关的信息，这些也是可用于类似但不要求必须是完全相同的应用。

图6-2 [Ara89]图示说明了域分析过程的关键输入和输出。应该调查领域知识的来源以便于确定可以在整个领域内复用的对象。

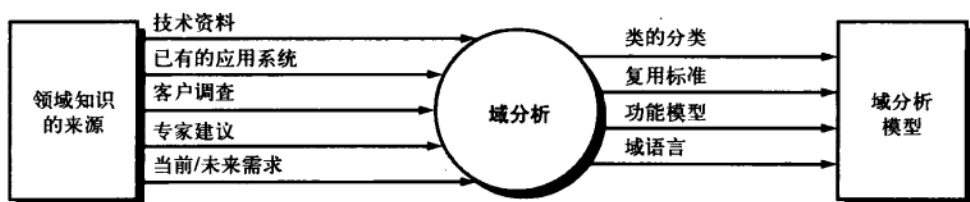


图6-2 域分析的输入和输出

SAFEHOME

域分析

[场景] Doug Miller的办公室，在销售业务会议之后

[人物] Doug Miller软件工程师经理，Vinod Raman 软件工程团队成员

[对话]

Doug: 我需要你做一个特殊的项目，Vinod。我将会把你从需求收集会议中抽出来。

Vinod (皱眉不悦): 太糟了。这可行吗……我正在从中获取了一些需求。出什么事啦？

Doug: Jamie和Ed将接管你的工作。不管怎样，市场部坚持要我们在第一次发布的SafeHome中交付具有互联网能力的家庭安全功能。我们一直紧张地为此工作着……没有足够的时间和人力，所以我们马上要解决PC机接口和Web接口两个问题。

Vinod (看上去很疑惑): 我不知道原先设定的计划……我们甚至还没有完成需求收集。

Doug (无精打采地微笑): 我知道，但时间太紧了我决定马上和市场部开始战略合作……无论如何，一旦从所有需求收集会议上获得信息，我们将重新审视任何不确定的计划。

Vinod: 好的，会发生什么事？你要我做什么事吗？

Doug: 你知道“域分析”吗？

Vinod: 略知一些。在建立应用系统时为做相同工作的应用系统寻找相似的模式。如果可能，可以在工作中剥离这些模式并复用它们。

Doug: 我觉得用剥离这个词不太恰当，但你的意思基本是正确的。我想让你做的事是开始研究为控制像SafeHome这类系统的现存用户接口。我想你要组织一套模式和分析类，它们通常既能坐在房间里处理基于PC机的接口，也能处理通过互联网进入基于浏览器的接口。

Vinod: 把它们做成相同的可以节省时间……为什么不这么做呢？

① 不要认为因为有域分析员在工作，软件工程师就不需要理解应用问题的领域。软件团队的每个成员都应该在一定程度上了解软件将要工作的领域。

Doug: 哦……有你这种想法的人真是非常好。整个核心要点是如果能识别出两种接口，采用相同的接口等，我们就节省了时间和人力。这些正是市场部坚持的。

Vinod: 那你想要什么？类、分析模式还是设计模式？

Doug: 所有的。非正式地说这就是关键所在。我就是想稍早一些开始我们内部分析和设计工作。

Vinod: 我将在类库中看看我们得到了哪些。我也会使用几个月前我读的一本书中的模式模板。

Doug: 太好了，继续工作吧。

……分析容易使人灰心丧气，全都是非常复杂的人际关系，不确定的和困难的东西。总而言之，分析让人着迷。一旦沉迷，原来轻松构建系统的快乐将难以令你满足。——Tom DeMarco

在描述需求模型时常用哪些不同的观点？

“我们为什么要构建模型？为什么不直接构建系统本身？答案是我们可以按照如下方式构建模型：突出或强调某些关键的系统特征，同时削弱系统的其他方面。”——Ed Yourdon

6.1.4 需求建模的方法

一种考虑数据和处理的需求建模方法被称作结构化分析，其中处理将数据作为独立实体加以转换。数据对象建模定义了对象的属性和关系，操作数据对象的处理建模应表明当数据对象在系统内流动时处理如何转换数据。

需求建模的第二种方法称做面向对象的分析，这种方法关注于定义类和影响客户需求的类之间的协作方式。UML和统一过程（第2章）主要是面向对象的分析方法。

尽管本书中我们提出的需求模型综合了两种方法的特点，但是软件团队往往选择一种方法并排斥另一种方法中的所有表示手段。问题不是哪一种方法最好，而是怎么组合这些表示手段才能够为利益相关者提供最好的软件需求模型和过渡到软件设计的最有效方法。

如图6-3所示需求模型的每个元素表示源自不同观点的问题。基于场景的元素表述用户如何与系统和使用软件时出现的特定活动序列进行交互。基于类的元素建模于系统操作的对象，应用在这些对象间影响操作和对象间关系（某层级）的操作，以及定义的类间发生的协作。行为元素描述了外部事件如何改变系统或驻留在系统里的类的状态。最后，面向流的元素表示信息转换的系统，描述了数据对象在流过各种系统功能时是如何转换的。

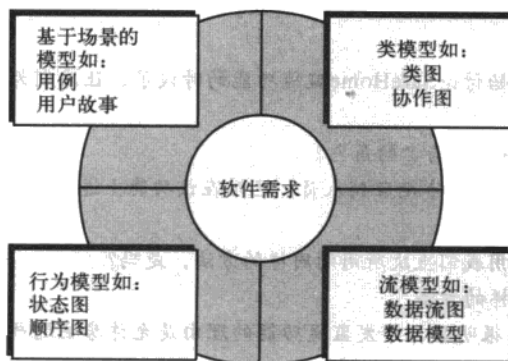


图6-3 需求模型的元素

需求建模导出每个建模元素的派生类。然而，每个元素（即用于构建元素和模型的图表）的特定内容可能因项目而异。就像我们在本书中多次提到的那样，软件团队必须想办法保持模型的简单性。只有那些为模型增加价值的建模元素才能使用。

6.2 基于场景建模

尽管可以用多种方式度量基于计算机的系统或产品的成果，用户的满意度仍是其中最重要的。如果软件工程师了解最终用户（和其他参与者）希望如何与系统交互，软件团队将能够更好地、更准确地刻画需求特征，完成更有针对性的分析和设计模型。因此，使用UML^①需求建模，将从开发用例、活动图和泳道图形式的场景开始。

6.2.1 新建初始用例

“[用例]只是帮助定义系统之外（参与者）存在什么以及系统应完成什么（用例）。”——
Ivar Jacobson

ADVICE
在某些情况下，用例成为最主要的需求工程机制，但是这并不意味着你应该放弃适用的其他的建模方法。

Alistair Cockburn刻画了一个名为“合同行为”的用例[Coc01b]。我们在第5章讨论的“合同”定义了一个活动者^②使用基于计算机的系统完成某个目标的方法。本质上用例捕获了信息的产生者、使用者和系统本身之间发生的交互。在本节，我们研究如何开发用例，这是需求建模活动的一部分^③。

第5章中我们已经提到，用例从某个特定参与者的角度出发，采用简明的语言描述一个特定的使用场景。但是我们如何知道：（1）编写什么？（2）写多少？（3）编写说明应该多详细？（4）如何组织说明？如果想让用例像一个需求建模工具那样提供价值，那么必须回答这些问题。

编写什么？两个首要的需求工程工作是起始和导出，它们提供了开始编写用例所需要的信息。运用需求收集会议、QFD和其他需求工程机制确定利益相关者，定义问题的范围，说明整体的运行目标，建立优先级顺序，概述所有已知的功能需求，描述系统将处理的信息（对象）。

开始开发用例时，应列出特定参与者执行的功能或活动。这些可以从所需系统功能的列表通过与利益相关者交流，或通过评估活动图（作为需求建模中的一部分而开发）获得（见6.3.1节）。

SAFEHOME

开发另一个初始的用户场景

[场景] 会议室，第二次需求收集会议中。

[人物] Jamie Lazar和Ed Robbins，软件团队成员；Doug Miller，软件工程经理；三个市场营销人员；一个产品工程代表；一个会议主持人。

[对话]

主持人：现在是开始讨论SafeHome监视功能的时候了，让我们为访问监视功能开发一个用户场景。

Jamie：谁在其中扮演参与者的角色？

主持人：我想Meredith（市场营销人员）已经在该功能上进行了一些工作，你来试试这个角色吧。

Meredith：你想采用我们上次所用的同样的方法，是吗？

主持人：是的，同样的方法。

Meredith：好的，很明显，开发监视功能的理由是允许房主远距离检查宅舍、记录并回放捕获的录像……就是这样。

① UML是本书通篇使用的建模符号。附录1为那些不熟悉UML基本符号的读者提供了简要指南。

② 参与者不是一个确定的人员，而是人员（或设备）在特定的环境内所扮演的一个角色，参与者“呼叫系统并由系统提供一种服务”[Coc01b]。

③ 用例是用户接口的分析建模中特别重要的一部分，将在第11章详细讨论接口分析。

Ed: 我们采用压缩的方法存储图像吗?

主持人: 好问题,但是我们现在先不考虑实现的问题,Meredith,你说呢?

Meredith: 好的,这样对于监视功能基本上就有两部分……第一部分是配置系统,包括布置建筑平面图——我们需要工具帮助房主做这件事,第二部分是实际的监视功能本身。因为布局是配置活动的一部分,我将重点集中在监视功能。

主持人 (微笑): 你说出了我想说的话。

Meredith: 哦……我希望通过电脑或通过Internet访问监视功能。我的感觉是Internet访问可能使用的频率更高一些。不管怎样,我希望能够在计算机上和控制面板上显示摄像机图像并移动某个摄像机镜头。在房屋平面设计图上可以选择指定摄像机,我希望可以有选择地记录摄像机输出和回放摄像机输出,我还希望能够使用特殊的密码阻止对某个或多个摄像机的访问。希望有支持小窗口显示形式的选项,即从所有的摄像机显示图像,并能够选择某一个放大。

Jamie: 那些称做缩略视图。

Meredith: 对,然后我希望从所有摄像机获得缩略视图。我也希望监视功能的接口和所有其他的SafeHome接口有相同的外观和感觉。

主持人: 干得好,现在,让我们更详细地讨论这个功能……

上面SafeHome框中讨论的SafeHome住宅监视功能(子系统)确定了如下由参与者房主执行的功能(简化列表):

- 选择将要查看的摄像机。
- 提供所有摄像机的缩略视图。
- 在计算机的窗口中显示摄像机视图。
- 控制某个特定摄像机的镜头转动和缩放。
- 可选择地记录摄像机的输出。
- 回放摄像机的输出。
- 通过Internet访问摄像机监视功能。

随着和利益相关者(扮演房主的人)更多地交谈,需求收集团队为每个标记的功能开发用例。通常,用例首先用非正式的描述性风格编写。如果需要更正式一些,可以使用类似于第5章中提出的某个结构化的形式重新编写同样的用例,在本节的后面将重新生成。

为了举例说明,考虑“通过互联网访问摄像机监视设备—显示摄像机视图(access camera surveillance-display camera views, ACS-DCV)”功能,扮演参与者房主的利益相关者可能会编写如下说明:


用例: 通过互联网访问摄像机监视设备—显示摄像机视图(ACS-DCV)

参与者: 房主

如果我位于远方,我可以使用任何计算机上合适的浏览器软件登录SafeHome产品网站。输入我的用户身份证号和两级密码,一旦被确认,我可以访问已安装的SafeHome系统的所有功能。为取得某个摄像机视图,从显示的主功能按钮中选择“监视”,然后选择“选取摄像机”,将会显示房屋的平面设计图,再选择感兴趣的摄像机。另一种可选方法是,通过选择“所有摄像机”,可以同时从所有的摄像机查看缩略视图快照。当选择了某个摄像机时,可以选择“查看”,然后以每秒一帧速度显示的图像就可以在窗口中显示。如果希望切换摄像机,选择“选取摄像机”,原来窗口显示的信息消失,并且再次显示房间的平面设计图,然后就可以选择感兴趣的摄像机,以便显示新的查看窗口。

描述性用例的一种表达形式是通过用户活动的顺序序列表现交互，每个行动由声明性的语句表示。再以ACS-DCV功能为例，我们可以写成：

用例：通过互联网访问摄像机监视设备—显示摄像机视图（ACS-DCV）

 用例可以有许多“软件”过程中，我们感兴趣的是迭代和风险驱动过程。——Geri Schneider and Jason Winters

参与者：房主

1. 房主登录SafeHome产品网站。
2. 房主输入用户身份证号。
3. 房主输入两个密码（每个至少八个字符长度）。
4. 系统显示所有的主要功能按钮。
5. 房主从主要功能按钮中选择“监视”。
6. 房主选择“选取摄像机”。
7. 系统显示房屋的平面设计图。


8. 房主从平面设计图中选择某个摄像机图标。
9. 房主选择“查看”按钮。
10. 系统显示一个由摄像机编号确定的查看窗口。
11. 系统在查看窗口内以每秒一帧的速度显示视频输出。

重要的是注意到，这个连续步骤的陈述没有考虑其他可能的交互（描述更加自由随意而且确实表达了一些其他选择）。这种类型的用例有时被称作主场景[SCH98]。

6.2.2 细化初始用例

为全面理解用例描述功能，对交互操作给出另外的描述是非常有必要的。

因此，主场景中的每个步骤将通过如下提问得到评估[SCH98]：

 在开发场景用例时，如何检查动作的可迭过程？

- 在这一状态点，参与者能进行一些其他动作吗？
- 在这一状态点，参与者有没有可能遇到一些错误的条件？如果有可能，这些错误会是什么？
- 在这一状态点，参与者有没有可能遇到一些其他的行为（如由一些参与者控制之外的事件调用）？如果有，这些行为是什么？

这些问题的答案导致创建一组次场景（Secondary Scenarios），次场景属于原始用例的一部分但是表现了可供选择的行为。例如，考虑前面描述的主场景的第6步和第7步：

6. 房主选择“选取摄像机”。
7. 系统显示房屋的平面设计图。

在这一状态点上，参与者能进行一些其他动作吗？答案是肯定的。参考无障碍解说方式，参与者可以选择同时查看所有摄像机的缩略视图。因此，一个次场景可能是“查看所有摄像机的缩略视图”。

在这一状态点，参与者有没有可能遇到一些错误的条件？作为基于计算机的系统操作，任何数量的错误条件都可能发生。在该上下文内，我们仅仅考虑在第6步和第7步中说明的活动的直接错误条件，问题的答案还是肯定的。带有摄像机图标的房屋平面图可能还没有配置，这样选择“选取摄像机”就导致错误的条件：“没有为该房屋配置平面设计图”^①。该错误条件就成为一个次场景。

在这一状态点，参与者有没有可能遇到一些其他的行为？问题的答案再一次是肯定的。当第6步和第7步发生时，系统可能遇到报警。这将导致系统显示一个特殊的报警通知（类型、地

^① 在该例子中，另一个参与者，系统管理员将必须配置平面设计图、安装并初始化（如分配设备编号）所有的摄像头，并且通过系统平面设计图访问和测试每个摄像头能达到的特定作用。

点、系统动作)，并向参与者提供和报警性质相关的一组操作。因为这个次场景可以在所有的实际交互中发生，所以不会成为ACS-DCV用例的一部分。而且，将开发一个单独的用例——“遇到报警条件”——这个用例可以被其他用例引用。

前面段落描述的每种情景都是以客户用例的异常处理为特征的。一个异常处理描述了这样一种情景（既可能是失败条件或参与者选择了替代方案），它导致系统展示出某些不同的行为。

Cockburn[Coc01b]推荐使用“头脑风暴”推动合理完成每个用例一系列的异常处理。除了本节前面的部分提到了三个常规问题外，还应该研究下面的问题：

- 在这个用例中是否有某些具有“有效功能”的用例出现？包括引用确认功能，可能出现的出错条件。
- 在这些用例中是否有支持功能（或参与者）的应答失败？例如，一个用户动作是等待一个应答，但该功能已经应答超时了。
- 性能差的系统是否会导致无法预期或不正确的用户活动？例如，一个基于Web的接口应答太慢，导致用户在处理按钮上已经做了多重选择。这些选择队列最终不恰当地生成一个出错条件。

这里列出的扩展部分是由具有因果关系的提问和回答问题开发的，应用下面的标准使这些问题合理化[Co01b]：用例应该注明异常处理即如果软件能检测出异常所发生的条件就应该在检测出后马上处理这个条件。在某些情况下，异常处理可能拖累影响其他用例处理条件的开发。

6.2.3 编写正规用例

在6.2.1节表述的非正规用例对于需求建模常常是够用的。但是，当一个用例包括关键活动或描述一套具有大量异常处理的复杂步骤时，就会希望采用更为正规的方法。

在SafeHome框中显示的ACS-DCV用例依照了正规用例的典型描述要点。在以下的SafeHome框中：“情境目标”确定了用例的全部范围。“前提条件”描述在用例初始化前应该知道哪些信息。“起动”确定“用例开始”的事件或条件[Coc01b]。“场景”列出由参与者和恰当的系统应答所需要的特定活动。“异常处理”用于细化初始用例时没有涉及的情境（6.2.2节）。此外还可能包含其他的标题，并给出合理的自我解释。

SAFEHOME

监视的用例模板

用例：通过互联网访问摄像头监视——显示摄像头视图（ACS-DCV）。

迭代：2，最新更改记录：V.Raman 1月14日。

主要参与者：房主。

情境目标：从任何远程地点通过互联网查看遍布房间的摄像头输出。

前提条件：必须完整配置系统；必须获得正确的用户身份证号和密码。

起动：房主在远离家的时候决定查看房屋内部。

场景：

1. 房主登录SafeHome产品网站。
2. 房主输入他或她的用户身份证号。
3. 房主输入两个密码（每个都至少有8个字符的长度）。
4. 系统显示所有的主要功能按钮。
5. 房主从主要功能按钮中选择“监视”。
6. 房主选择“选取摄像头”。

7. 系统显示房屋的平面设计图。
8. 房主从房屋的平面设计图中选择某个摄像头的图标。
9. 房主选择“视图”按钮。
10. 系统显示一个由摄像头编号确定的视图窗口。
11. 系统在视图窗口中以每秒一帧显示视频输出。

异常:

1. 身份证号或密码不正确或无法确认——参看用例:“确认身份证号和密码”。
2. 没有为该系统配置监视功能——系统显示恰当的错误消息;参看用例:“配置监视功能”。
3. 房主选择“查看所有摄像头的缩略视图快照”——参看用例:“查看所有摄像头的缩略视图快照”。

4. 平面设计图不可用或未配置——显示恰当的错误的消息,参看用例:“配置平面设计图”。

5. 遇到报警条件——参看用例:“遇到报警条件”。

优先级: 必须在基础功能之后实现中等优先级。

何时可用: 第三个增量。

使用频率: 中等频率。

使用方式: 通过基于个人计算机的浏览器和互联网连接到SafeHome网站。

次要参与者: 系统管理员, 摄像头。

次要参与者的使用方式:

1. 系统管理员: 基于个人计算机的系统。
2. 摄像机: 无线连接。

未解决的问题:

1. 有什么机制保护SafeHome产品的雇员在未授权的情况下能使用该功能?
2. 足够安全吗? 黑客入侵该功能将使最主要的个人隐私受侵。
3. 在给定摄像机视图所要求的带宽下, 可以接受通过互联网的系统响应吗?
4. 当可以使用高带宽的连接时, 能开发出比每秒一帧更快的视频速度吗?

WebRef

什么时候结束编写用例? 关于该话题有价值的论述, 可以参阅 ootips.org/use-cases-done.html。

在很多情况下, 不需要创建图形化表示的用户场景。然而, 尤其当场景比较复杂时图形化的表示更有助于理解。就如我们在本书前面所说提到的, UML的确提供了图形化表现用例的能力。图6-4为SafeHome产品描述了一个初步的用例图, 每个用例由一个椭圆表示。仅在本节中详细讨论了ACS-DCV。

每种建模注释方法都有其局限性, 用例方法也不例外。和其他描述形式一样, 用例几乎和它的作者(们)一样好。如果描述不清晰, 用例可能会误导或有歧义。一个用例关注功能和行为需求, 一般不适用于非功能需求。对于必须特别详细和精准的需求建模情景(例如安全关键系统), 用例方法就不够用了。

然而, 软件工程师会遇到的所有场景中的绝大多数情境都适用基于场景建模。如果开发得当, 用例作为一个建模工具能提供很重大的好处。

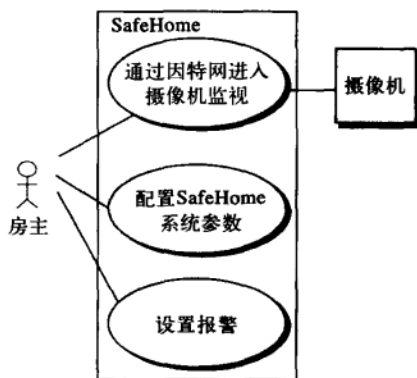


图6-4 SafeHome系统的初步用例图

6.3 补充用例的UML模型

很多基于文本的需求建模情景，甚至和用例一样简单，不能简明扼要地传递信息。在这种情况下，你应能从大量的UML图形模型中进行选择。

6.3.1 开发活动图

KEY POINT
一个UML活动图表现了实施某功能时发生的动作和判定。

UML活动图在特定场景内通过提供迭代流的图形化表示来补充用例。类似于流程图，活动图使用两端为半圆形的矩形表示一个特定的系统功能，箭头表示通过系统的流，判定菱形表示判定分支（标记从菱形发出的每个箭头），实心水平线意味着并行发生的活动。ACS-DCV用例的活动图如图6-5所示。应注意到活动图增加了额外的细节，而这些细节是用例不能直接描述的（是隐含的）。例如，用户尽可以尝试有限次数地输入用户身份证号（ID）和密码。这可以通过“提示重新输入”的判定菱形来体现。

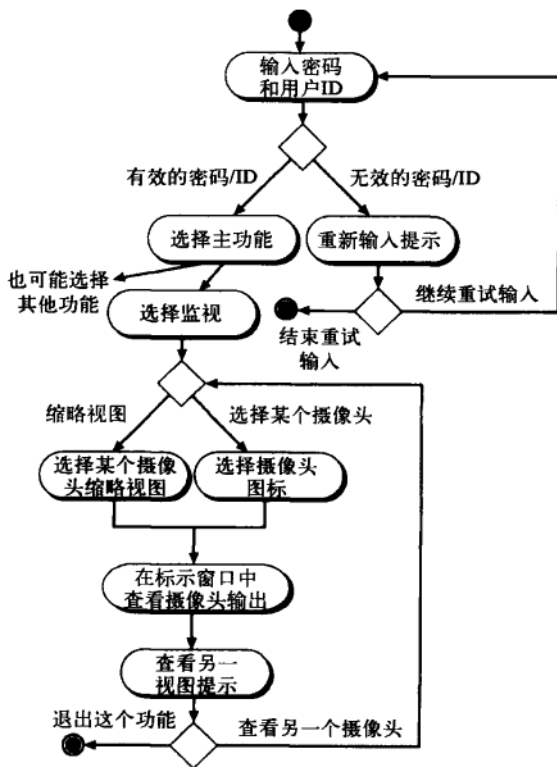


图6-5 通过互联网进入摄像机监视并显示摄像机视图功能的活动图

KEY POINT
UML泳道图表现了活动流和一些判定，并指明由哪个参与者实施。

6.3.2 泳道图

UML泳道图是活动图的一种有用的变形，可让建模人员表示用例所描述的活动流，同时指示哪个参与者（如果在某个特定用例中涉及了多个参与者）或分析类（本章后面章节会讨论）是由活动矩形所描述的活动来负责。职责由纵向分割图的并行条表示，就像游泳池中的泳道。

三种分析类（房主、摄像机和接口）对于在图6-5所表示的活动图中的情景具有直接或间接的责任。参看图6-6，重新排列活动图，与某个特殊分析类相关的活动按类落入相应的泳道中。例如，接口类表示房主可见的用户接口。活动图标记出对接口负责的两个提示——“提示重新输入”和“提示另一视图”。这些提示以及与此相关的判定都落入了接口泳道。但是，从该泳道发出的箭头返回到房主泳道，这是因为房主的活动在房主泳道中发生。

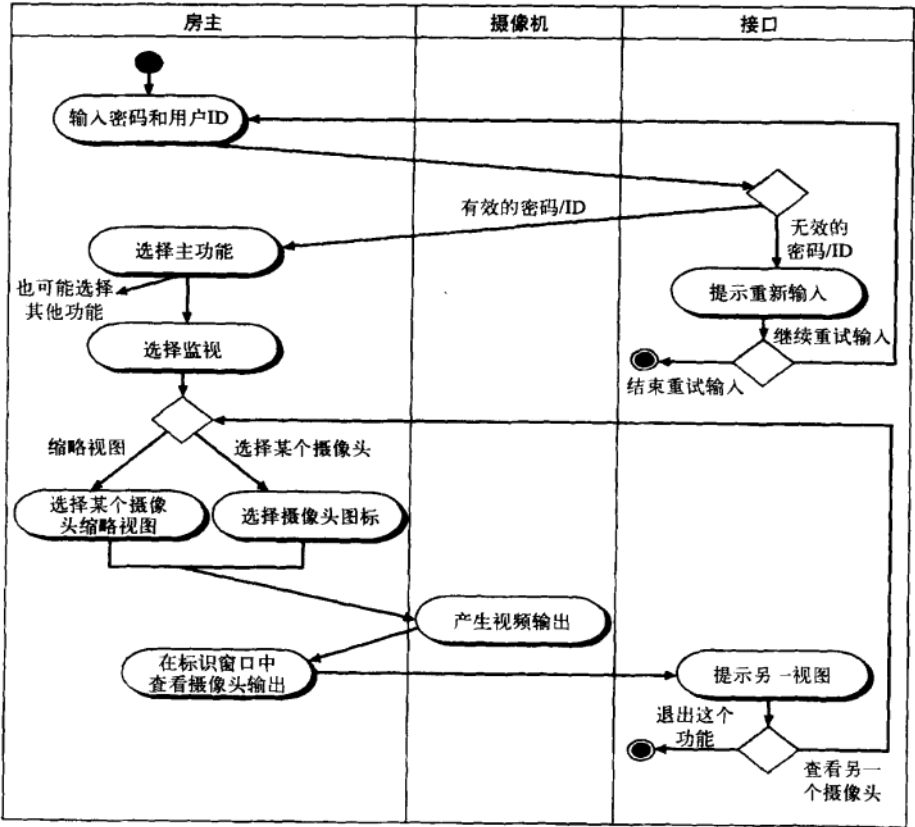


图6-6 通过互联网进入摄像机监视并显示摄像机视图功能的泳道图

“好的模型引导你的思考，而坏的模型会误导思考。”——Brian Marick

伴随着活动图和泳道图，面向过程的用例表示各种参与者行使一些特定功能（或其他处理步骤），以便满足系统需求。但是需求的过程视图仅表示系统的单一维度，在6.4节，我们将考察信息空间，并提供如何表达数据需求。

6.4 数据建模概念

WebRef
在 www.data-model.org 上可以找到数据建模的有用信息。

如果软件需求包括建立、扩展需求，或者具有数据库的接口，或者必须构建和操作复杂的数据结构，软件团队可以选择建立一个数据模型作为全部需求建模的一部分。软件工程师或分析师需要定义在系统内处理的所有数据对象，数据对象之间的关联以及其他与此相关的信息。实体关系图（ERD）描述了这些问题并提供了在一个应用项目中输入、存储、转换和产生的所有数据对象。

6.4.1 数据对象

数据对象如何在一个应用系统的环境内表现自己？

KEY POINT

数据对象是由计算机软件处理的任意复合信息的表示。

数据对象是必须由软件理解的复合信息表示。复合信息是指具有若干不同的特征或属性的事物。因此，“宽度”（单个的值）不是有效的数据对象，但是“维度”（包括宽度、高度和深度）可以被定义为一个对象。

数据对象可能是外部实体（例如产生或使用信息的任何东西），事物（例如报告或显示），偶发事件（例如电话呼叫）或事件（例如警报），角色（例如销售人员），组织单位（例如财务部），地点（例如仓库）或结构（例如文件）。例如，一个人或一部车可以被认为是数据对象，在某种意义上它们可以用一组属性来定义。数据对象描述包括了数据对象及其所有属性。

数据对象只封装数据——在数据对象中没有操作数据的引用^①。因此，数据可以表示为如图6-7所示的一张表，表头反映了对象的属性。在这个例子中，car是通过生产商、车型、标识号、车体类型、颜色和车主定义的。该表的主体表示了数据对象的特定实例。例如，Chevy Corvette牌的车是数据对象car的一个实例。

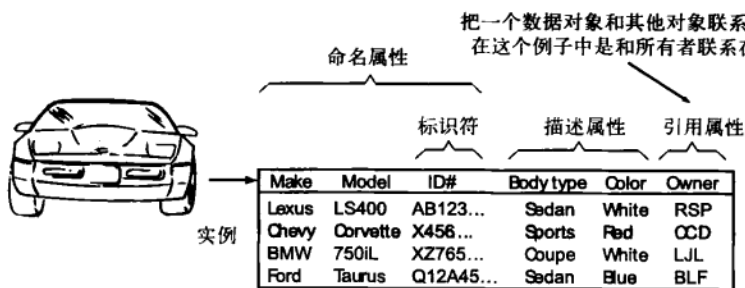


图6-7 数据对象的表格表示

6.4.2 数据属性

KEY POINT

属性命名某数据对象，描述其特征，并在某些情况下引用另一个对象。

数据属性定义了数据对象的性质，可以具有三种不同的特性之一。它们可以用来：（1）为数据对象的实例命名；（2）描述这个实例；（3）建立对另一个表中的另一个实例的引用。另外，必须把一个或多个属性定义为标识符——也就是说，当我们要找到数据对象的一个实例时，标识符属性成为一个“键”。在某些情况下，标识符的值是唯一的，但不是必须的。在数据对象汽车的例子中，标识号可以作为一个合理的标识符。

通过对问题环境的理解可以恰当地确定特定数据对象的一组属性。汽车的属性可以很好地用于汽车运输部门的应用系统，但这些属性对于汽车制造公司来说是无用的，汽车公司需要制造中的控制软件。在后一种情况下，汽车的属性可能也包括标识号、车体类型和颜色，但为了使汽车在制造中的控制环境下成为一个有用的对象，必须增加许多其他的属性（如内部代码、驱动系统类型、车内包装设计师、传动类型）。

WebRef

一种称为“规范化”的概念对于偏好全部使用数据建模的人非常重要，可以在 www.datamodel.org 中找到有用的信息。

① 这种区别将数据对象与面向对象方法中的“类”或“对象”区分开来（附录2）。

数据对象和面向对象类——它们是同一个东西吗？

当讨论数据对象时会出现一个常见的问题：数据对象和面向对象^①的类是同一个东西吗？答案是否定的。

数据对象定义了一个复合的数据项，也就是说合并一组独立的数据项（属性）并为数据项集合命名（数据对象名）。

一个面向对象类封装了数据属性，但对这些属性所定义数据的操作（方法）进行合并。另外，类的定义暗示了一个全面的基础设施，该基础设施是面向对象软件工程方法的一部分。类之间通过消息通信，它可以按层次关系组织，并为某个类的一个实例这样的对象提供继承特性。

6.4.3 关系

数据对象可以以多种不同的方式与另一个数据对象连接。考虑一下两个数据对象：person和car。这些对象可以使用图6-8a所示的简单标记表示。在person和car之间可以建立联系，因为这两个对象之间是相关的。但这个关系是什么呢？为确定答案，我们必须理解在将要构建的软件环境中人（在这里是指车主）和车的角色。我们可以用一组“对象/关系对”来定义相互的关系，例如：

- 拥有车的人。
- 汽车驾车投保人。

关系“拥有”和“驾车投保”定义了person和car之间的相关连接。图6-8b以图形



图6-8 数据对象间的关联关系

KEY POINT
关系指明数据对象相互“链接”的方式。

方式说明了这些对象/关系对，图6-8b标注的箭头提供了关联方向的重要信息，这一方向信息通常可以减少歧义或误解。

实体-关系图

对象/关系对是数据模型的基石。可以使用实体-关系图（ERD）^②图形化地表示这些对象/关系对。ERD最初是由Peter Chen[Che77]为关系数据库系统设计提出的，并由其他人进行了扩展。ERD标识了一组基本元素：数据对象、属性、关系以及各种类型的指示符。使用ERD的主要目的是表示数据对象及其关系。

已经介绍过基本的ERD符号，用带标记的矩形表示数据对象，用带标记的线连接对象表示关系。在ERD的某些变形中，连接线包含一个带有关系标记的菱形。使用各种指示基数和模态的特殊符号来建立^③数据对象和关系的连接。关于数据建模和实体关系图的更多信息，感兴趣的读者可以参考[Hob06]或[Sim05]。

① 如果读者不熟悉面向对象的内容和专有名词，应该参考附录2中介绍的简要指南。

② 尽管ERD仍然在某些数据库设计应用中使用，UML符号（附录1）在现在的数据设计中也很常用。

③ 对象关系对的基数（Cardinality）特指“一个[对象]出现的次数与另一个[对象]出现次数相关联”[Til93]。对于关系的模态（Modality）是指：如果没有明确发生关系的需要或者关系是可选的，那么关系的模态是0；如果关系必须出现一次，那么模态是1。

SOFTWARE TOOLS

数据建模

目的：数据建模工具为软件工程师提供表现数据对象、数据对象的特点和数据对象的关系的能力。主要用于大型数据库应用系统和其他信息系统项目，数据建模工具以自动化的方式创建全面的实体-关系图、数据对象字典以及相关模型。

机制：该类型的工具帮助用户描述数据对象及其关系。在某些情况下，工具使用ERD符号；有些情况下工具使用其他一些原理为关系建模。该类工具往往用来作数据库设计，还可通过为公共数据库管理系统（DBMS）生成数据库模式帮助创建数据库模型。

代表性工具：^①

AllFusion ERWin，由Computer Associates (www3.ca.com) 开发，辅助设计数据库的数据对象、恰当的结构和关键元素。

ER/Studio，由Embarcadero Software (www.embarcadero.com) 开发，支持实体-关系建模。

Oracle Designer，由Oracle Systems (www.oracle.com) 开发，为业务处理、数据实体和关系建模，并转化成可以生成完整应用系统和数据库的设计。

Visible Analyst，由Visible Systems (www.visible.com) 开发，支持包括数据建模在内的各种分析建模功能。

6.5 基于类的建模

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、类的职责协作者(CRC)模型、协作图和包。下面几小节中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。

6.5.1 识别分析类

“真正困难的课题是首先发现什么才是正确的对象（类）。”
——Carl Argila

如果环顾房间，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。

通过检查需求模型开发的使用场景，对系统开发的用例进行“语法解析”[Abb83]，我们可以开始进行类的识别。带有下划线的每个名词或名词词组可以确定为类，并将这些名词输入到一个简单的表中，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。

不过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。

分析类如何把自己表现为解决方案空间的元素？

① 这里给出的工具只是此类工具的例子，并不代表本书支持采用这些工具。大多数情况下，工具的名字由各自的开发者注册为商标。

- 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），定义了对应的类或与对象相关的类。

这种分类只是文献中已提出的大量分类之一^①。例如，Budd[Bud96]建议了一种类的分类法，包括数据产生者（源点）和数据使用者（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`”甚至用“`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“处理叙述”^②进行语法分析（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）。



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安全功能可以帮助房主在安装时配置安全系统，监控所有链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个人计算机设计和配置系统。为每个传感器分配一个编号和类型，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。

当识别出一个传感器事件时，软件激活装在系统上可发声的警报，由房东在系统配置活动中指定的延迟时间后，软件拨打监测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，直至达到电话接通。

房主通过控制面板、个人计算机或浏览器这些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……

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

① 另一个重要的分类是指定义实体、边界和控制类，在6.5.4节中讨论。

② “处理叙述”类似于用例的风格，但目标稍有不同。“处理叙述”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（导出）部分也会使用语法解析开发每个用例。

这个表应不断完善，直到已经考虑到了处理叙述中所有的名词。注意，我们称列表中的每一输入项为潜在的对象，在进行最终决定之前还必须对它们每一项深思熟虑。

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：

如何确定某个潜在类是否应该真的成为一个分析类？

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，在这种分析过程中的潜在类是有用的。

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，和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。

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
监测服务	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作出“接受或拒绝”不同的决定。（例如，如果每个房主都有个人密码或通过声音确认，Homeowner类有可能接受并满足特征1和2）。

6.5.2 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，属性是定义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跟踪职业棒球手的统计信息，类Player的属性与用于职业棒球手的养老系统中的属性是截然不同的。在前者，属性是与名字、位置、平均击球次数、担任防守百分比，从业年限、比赛次数等相关的。后者，某些属性具有相同的含义，另外一些属性将被替换（或引起争议），例如，平均工资、充分享受优惠权后的信用、所

“阶级斗争，一些阶级胜利了，一些阶级消灭了……”——毛泽东

KEY POINT

属性是在问题的环境下完整地定义类的对象集合。

选的养老计划、邮件地址等。

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“属于”类的“事物”。此外，每个类都应回答如下问题：什么数据项（组合项和/或者基本项）能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精化到基础级，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6-9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6-9中的数据项或属性。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。

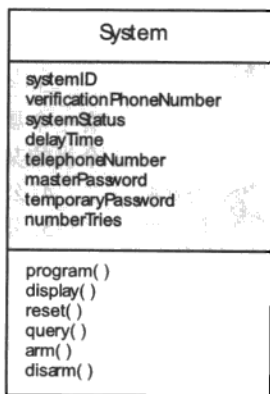


图6-9 System类的类图

ADVICE

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。

6.5.3 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和/或相关属性（6.5.5节）上的操作实现。因此，操作必须“理解”为类的属性和相关属性的性质。

在第一次迭代要导出一组分析类的操作时，可以再次研究处理叙述（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处理叙述中可以看到，“为传感器分配编号和类型”、“主密码用于激活和解除系统”，这些短语表明了一些事物：

- 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。

再进一步的研究，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。

类模型

SAFEHOME

[场景] Ed的小房间，开始进行需求建模。

[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SafeHome框中已有介绍）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，他或她可能从一个平面设计图进行选择。我已经定义了一个FloorPlan类，这个图在这里。

(他们查看图6-10。)

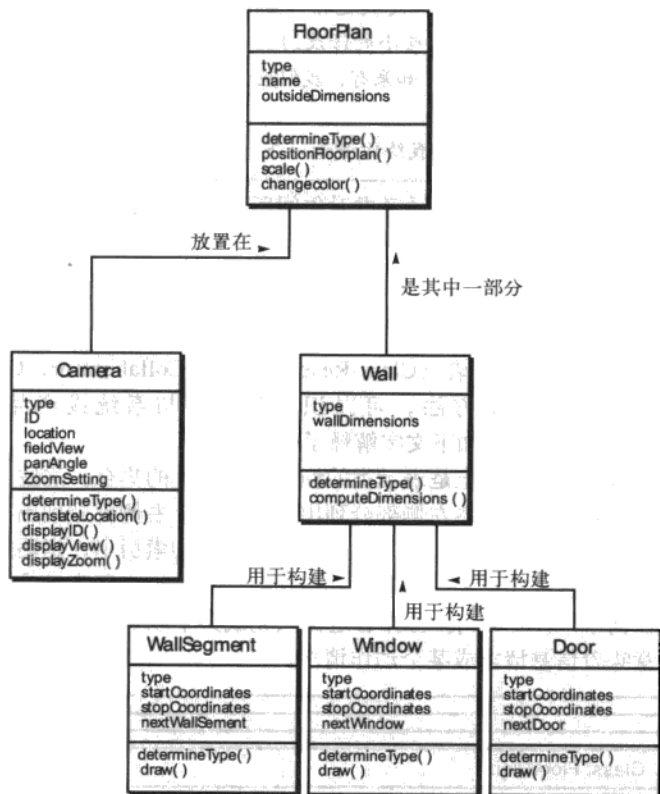


图6-10 FloorPlan类的类图

Jamie: 那么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6.5.5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。例如察看属性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编号而其他的没有呢？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, 当我们得到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疑问了。

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。

另外, 对于语法分析, 分析师能通过考虑对象间所发生的通信获得对其他的操作更为深入的了解。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, 探测到了更详实的信息。

6.5.4 类-职责-协作者建模

使用 CRC 卡的一个目的是早些舍弃、频繁舍弃, 并且低成本舍弃。事实上抽出一叠卡片要比改编大量源代码要容易得多。”
——C. Horstman

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模型可以使用真的或虚拟的索引卡, 意图是开发有组织表示的类。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。

Class: FloorPlan	
说明	
职责:	协作者:
定义住宅平面图名称/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	
显示摄像头的位置	Wall
	Camera

图6-11 CRC模型索引卡

FloorPlan类的一个简单CRC索引卡如图6-11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。在本章前面已经说明识别类和对象的基本原则。6.5.1节所说的类的分类可以通过如下

WebRef

关于这些类的类型的精彩讨论，可以参阅 www.theumlcafe.com/a0079.htm。

“对象可以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

为类分配职责时可以用来指导原则？

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应用程序（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。设计边界类的职责是管理实体对象对用户的表示方式。例如，一个称做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”[UML03]。也就是说，设计控制类可以管理（1）实体类的创建或更新；（2）当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。

职责。在6.5.2节和6.5.3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：

1. 智能系统应分布在所有类中以求最佳地满足问题的需求。每个应用系统都包含一定程度的智能，也就是系统内所含有它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一些事情（通常是适度集中），并提高系统的内聚性^①，这将提高软件的可维护性并减少变更的副作用影响。

为了确定是否恰当地分布智能系统，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^②。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，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

① 内聚性将在第8章设计概念中讨论。

② 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。

speed) 并且所有这些属性都必须在用户操纵游戏杆时更新和显示。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, 但是每个对象控制各自的显示。

协作。类有一种或两种方法实现其职责: (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; 或者 (2) 一个类可以和其他类协作。Wirfs-Brock和她的同事[Wir90]如下定义协作:

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。协作是客户和服务端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, 每个协作都和服务端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, **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。

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。

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6-12所示的聚合方式表示这些关系。

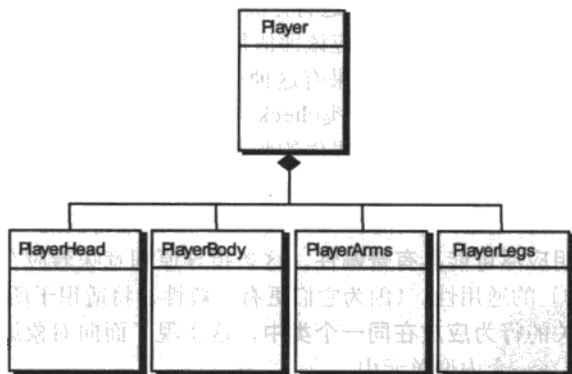


图6-12 复合聚合类

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** (除非视频游戏特别异常), 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**。

所有情况下，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6-11）。

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。“暗示着某个传感器是开启的”语句需要索引卡包含确认该暗示的职责（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。

4. 当令牌传递时，Sensor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编写结束。当评审完所有的用例，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。系统应该让我能够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的温度。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。

通过一个选择就能够针对各种情况设置整个房屋，第一个选择项是“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6-10)。

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。

6.5.5 关联和依赖

KEY POINT

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的联系数量关系。

在很多例子中, 两个分析类以某种方式相互联系着, 就如同彼此相互联系的两个数据对象 (6.4.3节)。在UML中, 这些联系被称作关联 (associations)。参考图6-10, 通过识别FloorPlan和另外两个类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6-10, 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6-13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^①。

① 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。

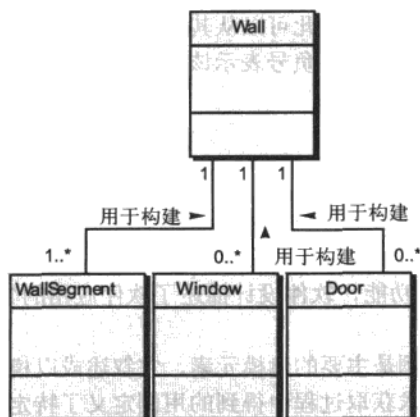


图6-13 多样性

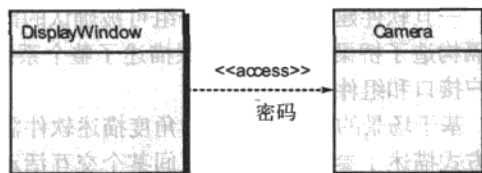


图6-14 依赖

什么是构造型？

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》）。

下面举例说明SafeHome监视系统中的简单依赖关系。一个Camera对象（本例中的服务器类）向一个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6-14表示，其中《access》意味着通过特定的密码控制使用摄像机的输出。

6.5.6 分析包

KEY POINT

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，分组打包后称其为分析包，并取一个有代表性的名。

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建成，并且已经生成大量的类。有一些类关注于游戏环境，即用户游戏时所看到的可视场景。诸如Tree、Landscape、Road、Wall、Bridge、Building、VisualEffect类等可能就属于游戏环境类。另一些类关注于游戏内的人物，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，可以由图6-15中的分析包表示这些类。

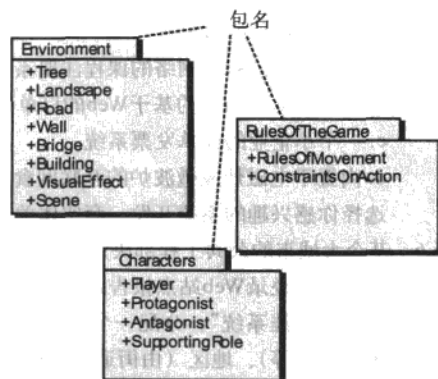


图6-15 包

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可以添加其他符号。负号表示该元素对其他包是隐藏的，#号只能由指定包中的类访问表示该元素。

6.6 小结

需求建模的目标是创建各种表现形式，用其描述什么是客户需求，建立生成软件设计的基础，一旦软件建立就能定义一组可被确认的需求。需求模型为系统级表示层和软件设计之间的间隔构造了桥梁。系统表示层描述了整个系统和业务功能，软件设计描述了软件应用的架构、用户接口和组件级的结构。

基于场景的模型从用户的角度描述软件需求。用例是主要的建模元素，它叙述或以模板驱动方式描述了参与者和软件之间某个交互活动。在需求获取过程中得到的用例定义了特定功能或交互活动的关键步骤。用例的形式化和详细程度各不相同，但是最终结果为所有的其他分析建模活动提供了必需的输入。还可以使用活动图说明场景，即一种类似于流程图的图形表现形式，描述在特定场景中的处理流。泳道图显示了如何给不同的参与者或类分配处理流。

数据建模常用于描述了软件构建或操作的信息空间。数据建模由所代表的数据对象开始，这些数据对象必须由软件所理解的信息组成。每个数据对象的属性得到识别，数据对象间的关系得到描述。

为了识别分析类，基于类的建模使用从基于场景和面向流的建模元素中导出信息。可以使用语法分析从文本叙述中提取候选类、属性和操作，并制定了用于定义类的标准。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建模方法定义类之间的层次、关系、关联、聚合和依赖。使用一种分析包方式进行分类和分组，从而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

- 6.1 有没有可能在分析模型创建后立即开始编码？解释你的答案，然后说服反方。
- 6.2 一个单凭经验的分析原则是：模型“应该关注在问题域或业务域中可见的需求”。在这些域中哪些类型的需求是不可见的？提供一些例子。
- 6.3 域分析的目的是什么？如何将域分析与需求模式概念相联系？
- 6.4 有没有可能不完成如图6-3所示的4种元素就开发出一个有效的分析模型？解释一下。
- 6.5 构建如下系统中的一个：
 - a. 你所在大学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实体关系图并说明数据对象、关系和属性。
- 6.6 某个大城市的公共工程部决定开发基于Web的跟踪和修补路面坑洼系统（PHTRS）。说明如下：

市民可以登录Web站点报告路面坑洼的地点和严重程度。当上报路面坑洼时，它被记入“跟踪和修补路面坑洼系统”，分配一个标识号，保存如下信息：街道地址、大小（比例从1到10）、位置（中央、路边等）、地区（由街道地址确定）以及修补优先级（由坑洼大小确定）。工作订单数据和每个坑洼有关联，数据包含坑洼位置和大小、维修组标识号、维修组内人员数量、分配的设备、修复耗时、坑洼状态（正在处理中、已修复、临时修复、未修复）、使用的填充材料数量以及修复成本

(从修复耗时、人员数量、材料和使用的设备计算)。最后,生成损失文件以便保存该坑洼所造成的损失报告信息,并包含公民的姓名、地址、电话号码、损失类型、损失金额。PHTRS是基于在线系统,可交互地进行所有的查询。

a. 为PHTRS系统画出UML用例图,你必须对用户和系统的交互方式做一些假设。

b. 为PHTRS系统开发一个类模型。

6.7 为6.5.4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。

6.8 为6.5题所选的产品或系统开发一组完整的CRC模型索引卡。

6.9 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?

6.10 什么是分析包?如何使用分析包?

推荐读物与阅读信息

用例是为所有需求建模方法服务的基础。有大量详细讨论这一主题的书,包括Rosenberga 和 Stephens (《Use Case Driven Object Modeling with UML:Theory and Practice》,Apress, 2007)、Denny (《Succeeding with Use Cases:Working Smart to Deliver Quality》,Addison-Wesley, 2005)、Alexsander 和 Maiden (《Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle》, Wiley, 2004)、Bittner和Spence (《Use Case Modeling》,Addison-Wesley, 2002)、Cockburn[Coc01b],以及其他参考资料请参考第5章和第6章。

数据建模为检验信息空间描述了非常有用的方法,相关书目有:Hoberman [Hob06]和Simsion 和 Witt[Sim05]提出了更合理全面的处理。另外,Allen和Terry(《Beginning Relational Data Modeling》, 2nd ed., Apress,2005)、Allen (《Data Modeling for Everyone》, Wrox Press, 2002)、Teorey和他的同事 (《Database Modeling and Design: Logical Design》, 4th ed., Morgan Kaufmann, 2005)、Carlis 和 Maguire (《Mastering Data Modeling》, Addison-Wesley, 2000) 给出了生成行业质量数据模型的详细指南。另一本有趣的书是Hay(《Data Modeling Patterns》, Dorset House, 1995)。描述了许多不同业务领域使用典型数据模型的模式。

UML建模技术可以应用于分析和设计,讨论这方面的书目有O'Docherty (《Object-Oriented Analysis and Design: Understanding System Development with UML 2.0》,Wiley, 2005)、Arlow 和Neustadt (《UML 2 and the Unified Process》, 2nd ed., Addison-Wesley,2005)、Roques (《UML in Practice》, Wiley, 2004)、Dennis 和他的同事 (《Systems Analysis and Design with UML 2.0》, Wiley, 2004)、Larman (《Applying UML and Patterns》, 2nd ed., Prentice-Hall, 2001)、Rosenberg 和Scott (《Use Case Driven Object Modeling with UML》, Addison-Wesley, 1999)。

互联网上有很多关于需求建模的信息。可以参考SEPA网站上有关分析建模的更新列表:
www.mhhe.com/engcs/compsci/pressman/professional/olc/ser.htm。