第 11 章

Software Engineering: A Practitioner's Approach, Eighth Edition

需求建模：行为、模式和 Web / 移动 App

要点浏览

概念：在这一章将学习需求建模的其他维度——面向行为建模、模式，以及开发 WebApp 时要考虑的特定需求分析问题。这里的每种模型都是第 9 章和第 10 章的补充。

人员：软件工程师（有时称为分析师）从各个利益相关者中提取需求进行建模。

重要性：软件工程师洞悉到软件需求的增长与来自不同需求模型维度的增长成正比。尽管可能没有时间，缺乏资源，无法采用第 9 ～ 11 章所建议的任何一种表示方法进行开发，但是软件工程师有必要认知每种不同的建模方式，这会给他提供审视问题的不同方法。继而他（和其他利益相关者）将能够更好地进入状态，更好地界定是否已把必须完成的需求真正描述清楚。

步骤：行为建模描述了系统及其类的状态，以及事件对这些类的影响。基于模式的建模利用现有领域的知识使得需求分析更为容易。WebApp 的需求模型特别适用于表述内容、交互操作、功能和配置相关的需求。

工作产品：可为需求建模选择大量不同的基于文本和图形的表示方法。每种表示方法都提供了一种或多种模型元素。

质量保证措施：必须评审需求建模产品的正确性、完备性和一致性。必须反映所有利益相关者的要求，为导出设计建立基础。

在第 9 章和第 10 章讨论了基于场景建模和基于类模型后，我们很自然会问："这些需求建模表示方法就足够了吗？"

唯一合理的回答是："要看情况而定。"

对于某种类型的软件，用例可能是唯一可行的需求建模表示方法。而对于其他类型的软件，则需要选择面向对象的方法开发基于类的模型。但在另外一些情形下，可能必须要对复杂应用软件需求做个检测：查看当数据对象在系统中移动时是如何转换的；查看作为外部事件后果的应用软件是如何工作的；查看现存知识领域能否解决当前问题；或者在基于 Web 的系统或移动系统和应用软件中，如何将内容和功能融合在一起，使得最终用户能够利用 WebApp 实现相关目标。

关键概念

分析模式
行为模型
配置模型
内容模型
事件
功能模型
交互模型
导航建模
顺序图
状态图
状态表达

11.1 生成行为模型

前面章节讨论的建模表示方法表达了需求模型中的静态元素，现在则是把它转换成系统或产品的动态行为的时候了。要实现这一任务，可以把系统行为表示成一个特定的事件和时间的函数。

行为模型显示了软件如何对外部事件或激励做出响应。要生成模型，分析师必须按如下步骤进行：（1）评估所有的用例，以保证完全理解系统内的交互顺序；（2）识别驱动交互顺序的事件，并理解这些事件如何与特定的对象相互关联；（3）为每个用例生成序列；（4）创建系统状态图；（5）评审行为模型以验证准确性和一致性。

提问　如何用模型表明软件对某些外部事件的影响？

在下面的几节中将讨论每个步骤。

11.2　识别用例事件

在第 9 章中，我们知道用例表现了涉及参与者和系统的活动顺序。一般而言，只要系统和参与者之间交换了信息就发生事件。事件应该不是被交换的信息，而是已交换信息的事实。

为了说明如何从信息交换的角度检查用例，让我们再来考察 SafeHome 安全功能中的一部分用例。

房主使用键盘键入 4 位密码。系统将该密码与已保存的有效密码相比较。如果密码不正确，控制面板将鸣叫一声并复位以等待下一次输入；如果密码正确，控制面板等待进一步的操作。

用例场景中加下划线的部分表示事件。应确认每个事件的参与者，应标记交换的所有信息，而且应列出任何条件或限制。

举个典型事件的例子，考虑用例中加下划线的“房主使用键盘键入 4 位密码”。在需求模型的环境下，对象 Homeowner[⊖]向对象 ControlPanel 发送一个事件。这个事件可以称作*输入密码*。传输的信息是组成密码的 4 位数字，但这不是行为模型的本质部分。重要的是注意到某些事件对用例的控制流有明显影响，而其他事件对控制流没有直接影响。例如，事件*输入密码*不会明显地改变用例的控制流，但是事件*比较密码*（从与事件“系统将该密码与保存的有效密码相比较”的交互中得到）的结果将明显影响到 SafeHome 软件的信息流和控制流。 203

一旦确定了所有的事件，这些事件将被分配到所涉及的对象，对象负责生成事件（例如，Homeowner 房主生成*输入密码*事件）或识别已经在其他地方发生的事件（例如，ControlPanel 控制面板识别*比较密码*事件的二元结果）。

11.3　状态表达

关键点　系统状态可以表现特定的外部可观察的行为，类的状态可以表现当前系统执行功能时的行为。

在行为建模中，必须考虑两种不同的状态描述：（1）系统执行其功能时每个类的状态；（2）系统执行其功能时从外部观察到的系统状态。

类状态具有被动和主动两种特征 [Cha93]。*被动状态*只是某个对象所有属性的当前状态。例如，类 Player 的被动状态（在第 10 章讨论的视频游戏应用程序中）将包含 Player 当前的 position 和 orientation 属性，以及和游戏相关的 Player 的其他特性（例如，用来显示 magic wishes remaining 的属性）。对象的*主动状态*指的是对象进行持续变换或处理时的当前状态。类 Player 可能具有如下的主动状态：*移动*、*休息*、*受伤*、*疗伤*、*被捕*、*失踪*等。事件（有时称为*触发器*）才能迫使对象做出从一个主动状态到另一个主动状态的转移。

⊖ 在这个例子中，我们假设每位与 SafeHome 交互的用户（房主）都拥有确认的密码，因此都是合法的对象。

下面将讨论两种不同的行为表现形式。第一种显示一个类如何改变基于外部事件的状态，第二种以时间函数的形式显示软件的行为。

分析类的状态图。UML 状态图[⊖]就是一种行为模型，该图为每个类呈现了主动状态和导致这些主动状态发生变化的事件（触发器）。图 11-1 举例说明了 SafeHome 安全功能中 ControlPanel 类的状态图。

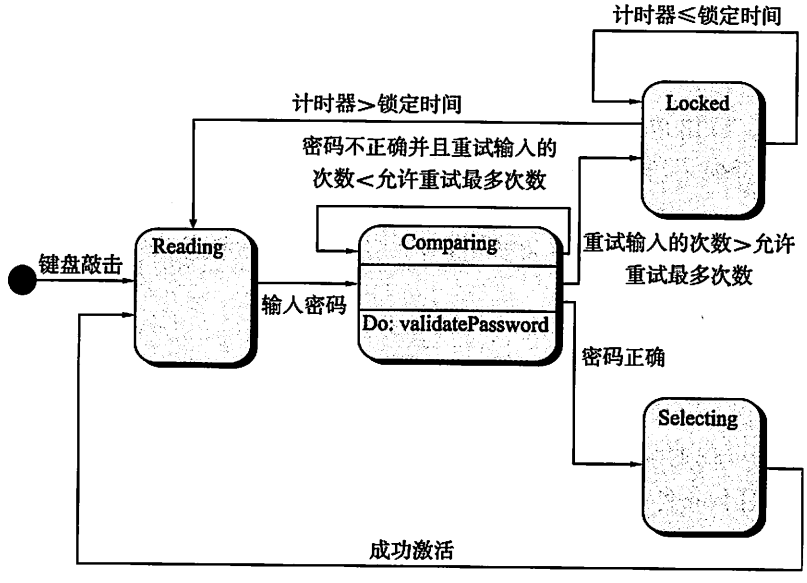

图 11-1　ControlPanel 类的状态图

图 11-1 中显示的每个箭头表示某个对象从一个主动状态转移到另一个主动状态。每个箭头上的标注都体现了触发状态转移的事件。尽管主动状态模型在提供对象的“生命历史”信息方面非常有用，但也能提供另外一些信息以便更深入地理解对象的行为。除了说明导致转移发生的事件外，分析师还可以说明守卫和动作 [Cha93]。守卫是为了保证转移发生而必须满足的一个布尔条件。例如，图 11-1 中从“读取”状态转移到“比较”状态的守卫可以由检查用例来确定：

204

```
if (password input = 4 digits) then compare to stored password
```

一般而言，转移的守卫通常依赖于某个对象的一个或多个属性值。换句话说，守卫依赖于对象的被动状态。

动作是与状态转移同时发生的，或者作为状态转移的结果，通常包含对象的一个或多个操作（职责）。例如，和输入密码事件（见图 11-1）相关联的动作是由 validatePassword() 操作的，该操作访问 password 对象并通过执行按位比较来验证输入的密码。

关键点　与不注明相关类表现行为的状态图不同，顺序图通过说明类如何从一个状态转移到另一状态来表现行为。

顺序图。第二种表现行为的方式在 UML 中称作顺序图，它表明事件如何引发从一个对象到另一个对象的转移。一旦通过检查用例确认了事件，建模人员就创建了一个顺序图，即用时间函数表现如何引发事件从一个对象流到另一个对象。事实上，顺序图是用例的速记版本。它表现了导致行为从一个类流

205

⊖ 如果读者对 UML 不熟悉，在附录 1 中有这些重要建模符号的简单介绍。

到另一个类的关键类和事件。

图 11-2 给出了 SafeHome 安全功能的部分顺序图。每个箭头代表了一个事件（源自一个用例）并说明了事件如何引导 SafeHome 对象之间的行为。时间纵向（向下）度量，窄的纵向矩形表示处理某个活动所用的时间。沿着纵向的时间线可以展示出对象的状态。

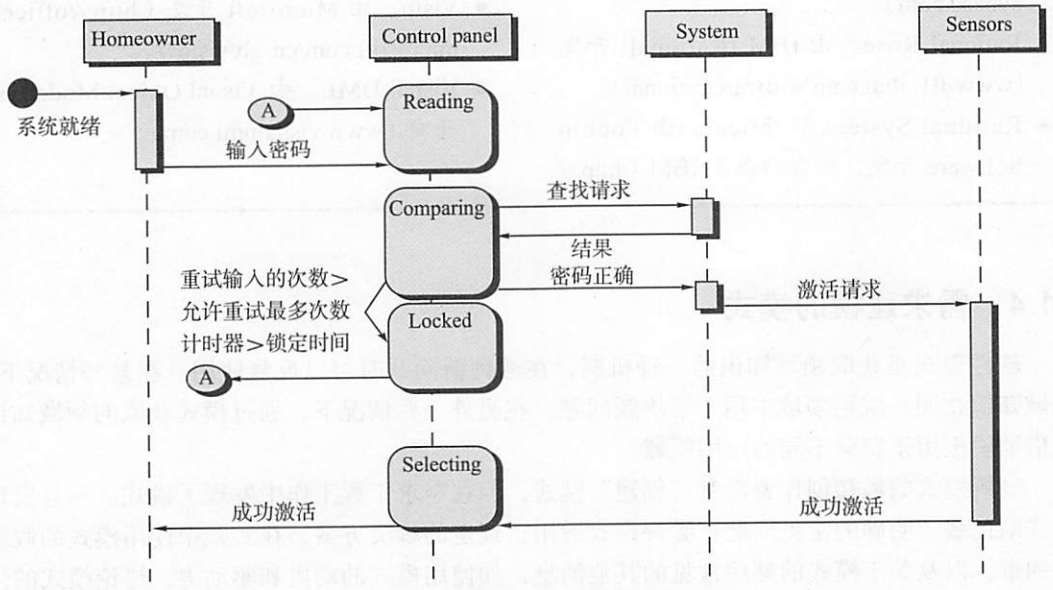

图 11-2　SafeHome 安全功能的顺序图（部分）

第一个事件*系统就绪*来自于外部环境并且把行为引导到对象 Homeowner。房主输入一个密码，*查找请求*事件发送到 System，它在一个简单的数据库中查找密码并向 ControlPanel（现在处在比较状态）返回（找到或没有找到）结果。有效的密码将促使系统产生*密码正确*事件，该事件通过*激活请求*事件激活传感器。最终，通过控制把*成功激活*事件返回到房主。

一旦完成了完整的顺序图，就把所有导致系统对象之间转移的事件整理为输入事件集合和输出事件集合（来自一个对象）。对于将要构建的系统而言，这些信息对于创建系统的有效设计非常有用。

206

软件工具　UML 中的通用分析建模

[目标] 分析建模工具可以使用 UML 语言开发基于场景的模型、基于类的模型和行为模型。

[机制] 这类工具支持构建分析模型所需的全部 UML 图（这些工具也支持设计建模）。除了制图，这类工具：（1）为所有的 UML 图执行一致性和正确性检查；（2）为设计和代码生成提供链接；（3）构建数据库，以便实现管理和评估复杂系统所需的大型 UML 模型。

[代表性工具][⊖]

如下工具支持分析建模所需的全部 UML 图：

- ArgoUML。开源工具（argouml.tigris.

⊖ 这里记录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。

org)。

- Enterprise Architect。由 Sparx Systems 开发(www.sparxsystems.com.au)。
- PowerDesigner。由 Sybase 开发(www.sybase.com)。
- Rational Rose。由 IBM (Rational) 开发(www-01.ibm.com/software/rational)。
- Rational System Architect。由 Popkin Software 开发,现在归属于 IBM(http://www-01.ibm.com/software/awdtools/systemarchitect/)。
- UML Studio。由 Pragsoft Corporation 开发(www.pragsoft.com)。
- Visio。由 Microsoft 开发(http://office.microsoft.com/en-gb/visio/)。
- Visual UML。由 Visual Object Modelers 开发(www.visualuml.com)。

11.4 需求建模的模式

软件模式是获取领域知识的一种机制,在遇到新问题时可以反复使用。在某些情况下,领域知识在同一应用领域中用于解决新问题。在另外一些情况下,通过模式获取的领域知识可借助模拟用于完全不同的应用领域。

分析模式的最初创作者没有“创建”模式,但在需求工程工作中发现了模式。一旦发现模式则记载“明确的常见问题:哪种模式适用,规定的解决方案,在实践中使用模式的假设和约束,以及关于模式的某些常见的其他消息,如使用模式的动机和驱动力,讨论模式的优缺点,参考在实践应用中某些已知的使用模式的样例”[Dev01]。

第 8 章引入了分析模式的概念并指明这些模式表示了某些应用领域(例如一个类、一个
207 功能或一个行为),当为这个领域的应用执行需求建模时会重复使用这些模式[⊖]。分析模式都存储在一个仓库中,以便软件团队的成员能够使用搜索工具找到并复用。一旦选择到合适的模式,就可以通过引用模式名称将其整合到需求模型中。

11.4.1 发现分析模式

需求模型由各种元素组成:基于场景(用例)、基于类(对象和类)和行为(事件和状态)。其中每个元素都是从不同的视角检查问题,并且每一个都提供一种发现模式的机会,可能发生在整个应用领域,或者发生在类似但不同的应用领域。

在需求模型的描述中最基本的元素是用例。在这个讨论环境下,一套连贯用例可以成为服务于发现一个或多个分析模式的基础。*语义分析模式*(Semantic Analysis Pattern, SAP)“描述了一小套连贯用例,这些用例一起描述了通用应用的基础”[Fer00]。

以下我们考虑要求控制和监控“实时摄像机”以及汽车临近传感器的软件用例。

用例:监控反向运动。

描述:当车辆安装了反向装置后,控制软件就能将后向摄像机的视频输入仪表板显示器。控制软件在仪表板显示器上叠加各种距离线和方向线,以便车辆向后运动时驾驶员能保持方向。控制软件还能监控临近传感器,以判定在车后方 10 英尺内是否有物体存在。如果临近传感器检测到某个物体在车后方 x 英尺内就会让车自动停止,这个 x 值由车辆的速度决定。

⊖ 在软件设计中,关于模式的进一步讨论见第 16 章。

在需求收集和建模阶段，本用例包含（在一套连贯用例中）各种将要精炼和详细阐述的功能。无论完成得如何精炼，建议用例要简单，但还要广泛地适用于 SAP，即具有基于软件的监控和在一个物理系统中对传感器和执行器的控制。在本例中，“传感器”提供临近信息和视频信息。“执行器”用于车辆的停止系统（如果一个物体离车辆很近就会调用它）。但是更常见的情况是发现大量的应用模式。 208

许多不同应用领域的软件需要监控传感器和控制物理执行器。所依照的分析模式描述了能广泛应用的通用需求。这个模式叫做 Actuator-Sensor（执行器 – 传感器），将用作 SafeHome 的部分需求模型，将在随后的 11.4.2 节讨论。

SafeHome　发现分析模式

[场景] 一个会议室，一个团队会议。

[人物] Jamie Lazar，软件团队成员；Ed Robbins，软件团队成员；Doug Miller，软件工程经理。

[对话]

Doug: SafeHome 项目有关传感器网络的需求建模进展如何啊？

Jamie: 对我来说传感器的工作有点新，但我认为我能掌控好。

Doug: 我们能帮上你什么忙吗？

Jamie: 如果我曾建立过类似的系统，会更容易些。

Doug: 当然。

Ed: 我考虑到在这种情况下我们要找到一个分析模式，帮助我们为这些需求建模。

Doug: 如果我们能发现正确的模式，我们可以避免重新发明轮子。

Jamie: 对我而言这太好了。那么我们怎么开始呢？

Ed: 我们有一个库包含大量的分析和设计模式。我们只需要检索出符合需求的模式。

Doug: 看来就这么干吧。你认为怎样，Jamie？

Jamie: 如果 Ed 能帮我开始，我今天就按此行动了。

11.4.2　需求模式举例：执行器 – 传感器[⊖]

SafeHome 安全功能需求之一是能监控安全传感器（例如，入侵传感器，火、烟或一氧化碳传感器，水传感器）。基于互联网的 SafeHome 还将要求能控制住所内安全摄像机的活动（例如摇摄、变焦）。这意味着 SafeHome 软件必须管理各种传感器和“执行器”（例如摄像机控制机制）。

Konrad 和 Cheng[Kon02] 建议将需求模式命名为**执行器 – 传感器**，它为 SafeHome 软件这项需求的建模提供有用的指导。执行器 – 传感器模式的简约版本最初是为如下的汽车应用开发的。

模式名：执行器 – 传感器。

目的：详细说明在嵌入系统中的各种传感器和执行器。 209

动机：嵌入系统包含各种传感器和执行器，这些传感器和执行器都直接或间接连接到一个控制单元。虽然许多传感器和执行器看上去完全不同，但它们的行为是相似的，足以构成

⊖ 本节采用 [Kon02] 的内容，已得到作者许可。

一个模式。这种模式显示了如何为系统指定传感器和执行器，包括属性和操作。执行器－传感器模式为被动式传感器使用拉机制（对消息的显示要求），为主动传感器使用推机制（消息广播）。

约束：

- 每个被动传感器必须通过某种方法来读取传感器的输入和表示传感器值的属性。
- 每个主动传感器必须能在其值发生变更时广播更新消息。
- 每个主动传感器应该能发送一个生命刻度，即在特定时间帧中发布状态信息，以便检测出错误动作。
- 每个执行者必须通过某种方法来调用由 ComputingComponent 计算构件决定的适当应答。
- 每个传感器和执行器应有实施检测其自身操作状态的功能。
- 每个传感器和执行器应能测试接收值或发送值的有效性，并且当值超出指定边界时能设定其操作状态。

适用性：对有多个传感器和执行器的任何系统都是非常有用的。

结构体：图 11-3 显示了执行器－传感器模式的 UML 类图，执行器、被动传感器和主动传感器是抽象类。这个模式中有四种不同的传感器和执行器。

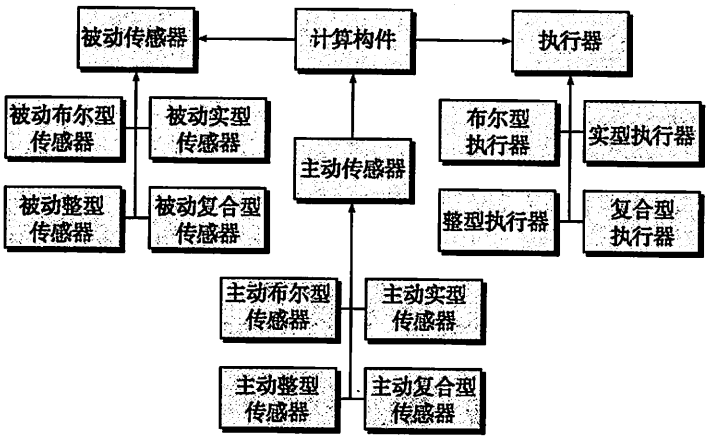

图 11-3　传感器和执行器的 UML 顺序图

传感器和执行器的常见类型为 Boolean、Integer 和 Real。就基本数据类型而言，复杂类是不能用值简单表示的，例如雷达设备。虽然如此，这些设备还是应该继承来自抽象类的接口，因为它们应该具备基本的功能（如查询操作状态）。

行为：图 11-4 描述了执行器－传感器例子的 UML 顺序图。这个例子可以应用于 SafeHome 功能，用以控制安全摄像机的调整（例如摇摄、变焦）。在读取或设置值时，ControlPanel 需要一个传感器（被动传感器）和一个执行器（摇动控制器）来进行以诊断为目的操作状态核查。名为 Set Physical Value 和 Get Physical Value 的消息不是对象间的信息，相反，它们描述了系统物理设备和相关软件对应项之间的交互活动。在图的下部（即在水平线以下），PositionSensor 报告操作状态为零。接着 ComputingComponent 发送位置传感器失败的错误代码给 FaultHandler 错误处理程序，它将决定这些错误对系统的影响，以及需要采取的措施。从传感器获得的数据经过计算得到执行器所需的应答。

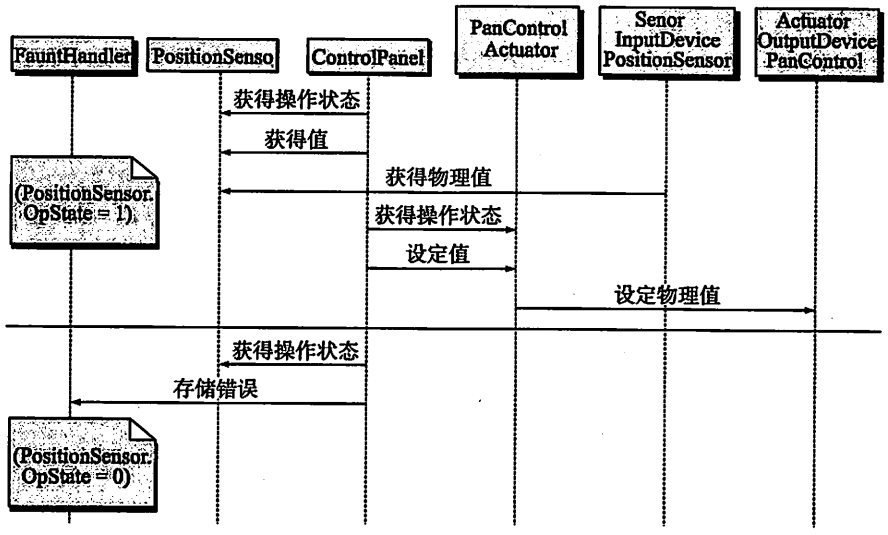

图 11-4　执行器 – 传感器模式的 UML 类图

210
~
211

参与者：模式“详细列举了包含在需求模式中的类或对象”[Kon02]，并描述了每个类或对象（图 11-3）的职责。简单列表如下：

- PassiveSensor abstract：定义被动传感器接口。
- PassiveBooleanSensor：定义被动布尔型传感器。
- PassiveIntegerSensor：定义被动整型传感器。
- PassiveRealSensor：定义被动实型传感器。
- ActiveSensor abstract：定义主动传感器接口。
- ActiveBooleanSensor：定义主动布尔型传感器。
- ActiveIntegerSensor：定义主动整型传感器。
- ActiveRealSensor：定义主动实型传感器。
- Actuator abstract：定义执行器接口。
- BooleanActuator：定义布尔型执行器。
- IntegerActuator：定义整型执行器。
- RealActuator：定义实型执行器。
- ComputingComponent：控制器的中心部分，它从传感器得到数据并为执行器计算所需的应答。
- ActiveComplexSensor：主动复合型传感器具备抽象类 ActiveSensor 的基本功能，但还需要指定额外更详细的方法和属性。
- PassiveComplexSensor：被动复合型传感器具备抽象类 PassiveSensor 的基本功能，但还需要指定额外更详细的方法和属性。
- ComplexActuator：复合型执行器具备抽象类 Actuator 的基本功能，但还需要指定额外更详细的方法和属性。

协作。本节描述的是对象和类之间如何进行交互活动，以及它们各自如何实现自身的责任。

- 当 ComputingComponent 需要更新 PassiveSensor 的值时，它询问传感器，通过发送

适当的消息请求传值。

- ActiveSensor 无需查询。这些对象和类启动传送过程，向计算机单元中传送传感器的值，使用适当方法设定在 ComputingComponent 中的值。对象和类在指定的时间帧发送至少一次生命刻度，以便用系统的时钟时间更新它们的时间戳。
- 当 ComputingComponent 需要设定执行器的值时，便会给执行器发送值。
- ComputingComponent 能使用适当的方法查询和设定传感器和执行器的操作状态。如果发现操作状态为零，就发送错误给 FaultHandler 错误处理程序，这个类包含处理错误消息的方法，比如启动更详细的恢复机制或者一个备份设备。如果不可恢复，系统只能使用传感器最后的已知值或者默认值。
- ActiveSensor 提供增加或消除地址或者组件地址范围的方法，一旦值发生变化，组件就能获得消息。

结果：

1. 传感器和执行器类有一个通用接口。
2. 只能通过消息访问类的属性，并且类决定是否接受这个消息。例如，如果执行器的值超过其最大值，执行器类就不可能接收消息，或者使用默认的最大值。
3. 系统潜在的复杂度得以降低是因为传感器和执行器有统一的接口。

需求模式的描述也能为其他相关的需求模式和设计模式提供参考。

11.5 Web / 移动 App 的需求建模[⊖]

Web/ 移动 App 开发者时常质疑我们所建议的需求分析观念。他们争辩道："开发过程终归必须使用敏捷方法，并且分析是非常耗时的。当我们需要设计和建立系统时就会减慢工作进度。"

需求分析确实很花时间，但是解决错误的问题甚至更花时间。每个 WebApp 和移动开发者的问题都不复杂，但是能否确保理解了问题或产品的需求？如果答案是毫不含糊的"是"，那么可以跳过需求建模，但是如果答案是"否"，就应该实施需求建模。

11.5.1 多少分析才够用

Web/ 移动 App 需求建模的重要程度依据以下因素而定：（1）应用软件的规模和复杂度的差异；（2）利益相关者的人数（所做的分析能帮助识别不同来源的需求冲突）；（3）WebApp 团队的人数；（4）一起工作以前该开发团队成员的业务水平（所做的分析有助于达成对项目的共同理解）；（5）组织成功的程度直接依赖于应用软件的成功。

与前面相反的观点是，当项目规模更小、利益相关者更少、开发团队更有内聚力，同时应用软件并非十分重要时，采用更轻量级的分析方法是合理的。

虽然在开始设计前进行问题或产品需求的分析是个好主意，但并不见得所有分析一定在所有设计之前。事实上，应用软件中仅有特定部分的设计要求分析对应用软件有影响的需求。例如 SafeHome，对所有网站做符合要求的美学设计（版面布局、色彩框架等）不需要分析电子商务能力的功能需求。为提高交付成果，软件分析师只需要分析与递增式交付相关

⊖ 本节部分内容来自 Pressman 和 Lowe[Pre08]，已得到作者许可。

的设计工作部分。[⊖]

11.5.2　需求建模的输入

当 Web/ 移动 App 已经工程化时，可以采用第 5 章讨论的常规软件过程的敏捷版本。这个过程包含的沟通活动可以识别利益相关者和用户类别、业务环境、界定信息和可用目标、普通的 WebApp 需求和使用场景，这些都是能成为需求建模输入的信息。这些信息以自然语言的描述、概要大纲、草图以及其他非正式形式表达。

分析这类信息，并使用（所适用的）正式定义的表达模式构造它，接着生成更缜密的模型作为输出。需求模型提供了揭示问题真实结构的详细指导，并提供洞察其特性的解决方案。

在第 9 章介绍 SafeHome 的 ACS-DCV（摄像机监视）功能时这个功能看上去相当清晰，并描述了用例的部分细节（9.2.1 节）。然而，当再次审查用例时就可能发现丢失的信息、模棱两可的信息或不清晰的信息。

某方面的丢失信息自然会在设计阶段显露出来。例如功能键的特定布局、美学外观、快 214
照视图尺寸、摄像机观察点位置、房间地板平面图，或者如密码的最大和最小长度这些细节。这些方面的某些内容是由设计决定的（如按键布局），另一些内容是根本不能影响早期设计工作的需求（如密码长度）。

但是有些丢失的信息实际上可能会影响总体设计，更多的则与需求的实际理解相关。例如：

问题 1：通过 SafeHome 摄像机输出的视频分辨率是多少？

问题 2：如果正在被监控的摄像机遇到警报条件，会发生什么？

问题 3：系统如何操作能摇摄和变焦的摄像机？

问题 4：摄像机的视图上还应该提供哪些信息？（例如，位置？时间或日期？上一次访问？）

在用例的初始开发阶段都没有考虑到这些问题，但是这些问题的答案却对设计的不同方面有着很大影响。

虽然沟通活动提供了很好的理解基础，但是需求分析通过提供额外的解释会使理解更精确，因此这个结论是合理的。当把问题的结构描述成需求模型的部分内容时，新问题又出现了。这就是弥补理解差距的问题，（或者在某些情况下）实际上这些问题也帮助我们在第一时间发现了这些差距。

总之，在沟通活动中收集到的信息作为需求模型的输入，即从非正式的电子邮件到包括综合使用场景和产品规格说明书中详细项目概述的任何内容。

11.5.3　需求建模的输出

需求分析提供了严格规定的机制来描述和评估 App 的内容和功能、用户将遇到的交互模式，以及 Web/ 移动 App 所处的环境和基础设施。

每种特性都能表示成一套模型，这套模型允许以结构化方式分析 App 的需求。当特定模型很大程度上依赖于 App 的性质时，会有 5 种主要的模型类型：

- **内容模型**给出由 App 提供的全部内容，包括文本、图表、图像、视频和音频数据。 215

⊖ 如果 App 中的一部分设计会对其他部分产生影响，那么分析的范围就应当扩大。

- **交互模型**描述了用户与 App 采用了哪种交互方式。
- **功能模型**定义了将用于处理 App 内容并描述其他处理功能的操作，这些处理功能不依赖于内容，但却是最终用户所必需的。
- **导航模型**为 App 定义所有导航策略。
- **配置模型**描述 App 所在的环境和基础设施。

分析师可以使用描述模式开发每一种模型，这种常称为“语言”的描述模式考虑到其意图和结构，使工程团队的成员和相关利益者之间更容易进行沟通和评估。最终识别出关键问题列表（例如错误、遗漏、不一致性、供增强和更改的建议、澄清观点）并照此行动。

11.5.4　内容模型

内容模型包含结构元素，它们为 App 的内容需求提供了一个重要的视图。这些结构元素包含内容对象和所有分析类，在用户通过浏览器或移动设备与 App 交互时生成并操作用户可见的实体[⊖]。

内容的开发可能发生在 App 实现之前、App 构建之中，或者 App 投入运行以后的很长一段时间里。在每种情况下，它都通过导航链接合并到 App 的总体结构中。内容对象可能是产品的文本描述、描述新闻事件的一篇文章、抽取数据的图形表示（例如随时间波动的股票价格）、在体育运动事件中拍摄的一张动作照片、在一次论坛讨论中某个用户的回答、公司徽标的一种生动演示、一个演讲的简短视频，或者是一套演示幻灯片中的音频插曲。这些对象内容可能存储于分离的文件中，或从数据库中动态获得，也可能直接嵌入 Web 页中，显示在移动设备屏幕上。换句话说，内容对象是呈现给最终用户的具有内聚信息的所有条目。

通过检查直接或间接引用内容的场景描述，可以根据用例直接确定出内容对象。例如，建立在 www.safehomeassured.com 中支持 SafeHome 的 WebApp。用例选择 SafeHome 构件描述了需要购买 SafeHome 构件的场景并包含如下句子：

216　*我将能得到每种产品构件的说明和价格信息。*

内容模型必须具备描述内容对象构件的能力。在许多实例中，用一个简单的内容对象列表，并给出每个对象的简短描述就足以定义设计和实现所必需的内容需求。但是在某些情况下，内容模型可以从更丰富的分析中获得好处，即在内容对象之间和 WebApp 维护的内容层次中采用图形化方法表示其关系。

例如，考虑图 11-5 中为 www.safehomeassured.com 构件建立的*数据树* [Sri01]。该树表述了常用于描述某个构件的一种信息层次。不带阴影的长方形表示简单或复合数据项（一个或多个数据值），带阴影的长方形表示内容对象。在此图中，描述说明是由 5 个内容对象（有阴影的长方形）定义的。在某些情况下，随着数据树的扩展，其中的一个或多个对象将会得到进一步细化。

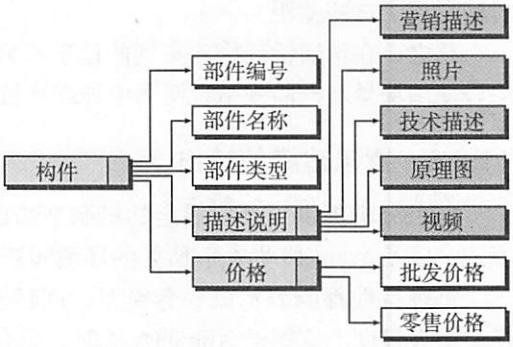

图 11-5　www.safehomeassured.com 构件的数据树

⊖ 分析类已在第 10 章讨论。

由多项内容对象和数据项组成的任何内容都可以生成数据树。开发数据树时要尽力在内容对象中定义层级关系，并提供一种审核内容的方法，从而在开始设计前发现遗漏的和不一致的内容。另外，数据树可以作为内容设计的基础。

11.5.5　WebApp 和移动 App 的交互模型

绝大多数 Web 和移动 App 都能使最终用户与 App 的功能、内容及行为之间进行“会话”。可以使用交互模型描述会话，这种交互模型由一种或多种元素组成：（1）用例；（2）顺序图；（3）状态图[⊖]；（4）用户界面原型。

在许多实例中，一套用例[⊜]足以描述在分析阶段的所有交互活动（在设计阶段引入更精确的细节）。然而当遇到复杂的交互顺序并包含多个分析类或多任务时，有时更值得采用严格图解方式描述它们。 217

用户界面的布局、介绍的内容、实现的交互机制以及用户与 WebApp 连接上的总体审美观，都与用户的满意度和 App 的整体成功密切相关。虽然有理由认为用户界面原型的创建是一种设计活动，但在创建分析模型期间执行用户界面原型的创建仍是一个好办法。对用户界面的物理表示评估得越早，就越有可能满足最终用户的需求。在第 15 章将详细讨论用户界面的设计。

因为 Web 和移动 App 的构造工具种类繁多，而且相对便宜、功能强大，因此最好使用这些工具创建界面原型。原型应该实现主要的导航链接，并采用同样的构造方式表现总体的屏幕布局。例如，为用户提供 5 种主要系统功能，当用户第一次进入 App 时，这些原型就应呈现出这些功能。如果要问将会提供图形链接吗？导航菜单会显示在哪里？用户会看到哪些其他信息？可以由原型回答以上类似的问题。

11.5.6　功能模型

许多 WebApp 都提供大量的计算和操作功能，这些功能与内容直接相关（既能使用又能生成内容）。这些功能常常以用户 –WebApp 的交互活动为主要目标。移动 App 更趋于关注和提供较有限的计算能力和操作功能的集合。但无论功能广度如何，都必须分析功能需求，并且在需要时也可以用于建模过程。

功能模型描述 WebApp 的两个处理元素，每个处理元素代表抽象过程的不同层次：（1）用户可观察到的功能是由 WebApp 传递给最终用户的；（2）分析类中的操作实现与类相关的行为。

用户可观察到的功能包括直接由用户启动的任何处理功能。例如，金融 WebApp 可以执行多种财务功能（例如支付抵押品的计算）。这些功能实际上可使用分析类中的操作完成，但是从最终用户的角度看，这些功能（更正确地说，这些功能所提供的数据）是可见的结果。

在抽象过程的更低层次，分析模型描述了由分析类操作执行的处理。这些操作可以操纵类的属性，并参与类之间的协作来完成所需要的行为。

不管抽象过程的层次如何，UML 的活动图都可用来表示处理细节。在分析阶段，仅在功能相对复杂的地方才会使用活动图。许多 WebApp 的复杂性不是呈现在提供的功能中，而是与可访问信息的性质以及操作的方式相关。 218

⊖ 使用 UML 符号为顺序图或状态图建模。

⊜ 用例的详细描述见第 9 章。

例如，在 www.safehomeassured.com 中，一个相对复杂的功能可以由一个用例来描述，这个用例名为*为我的空间中传感器的布局提供建议*。用户已经开发了一个可以监控空间的布局，在该用例中选择该布局，并要求传感器置于建议的位置。www.safehomeassured.com 用该布局的图形化表示方法回应，为推荐位置的传感器提供额外的信息。交互活动是非常简单的，某些内容有些复杂，然而它的底层功能非常复杂。系统必须承担地面布局的复杂分析，以便决定传感器的优化位置。必须检查房间面积、门窗位置，并协调这些传感器的功能和技术要求。这绝不是个小任务！我们使用一套活动图来描述这个用例所做的处理。

第二个例子是名为*控制摄像机*的用例。在这个用例中，交互活动相对简单，但存在潜在的复杂功能，这些“简单”操作需要通过访问互联网和远程设备进行复杂通信。当有多个已经授权的人同时想监视并且控制某个传感器时，便可能出现更复杂的情况，这与控制权协商有关。

图 11-6 为 takeControlOfCamera() 操作描绘了一幅活动图，它是*控制摄像机*用例中所用摄像机分析类的一部分。应该注意到程序流程中包括两项额外操作：requestCameraLock() 想为这个用户锁定摄像机；getCurrentCameraUser() 获取当前控制这台摄像机的用户名字。暂不考虑描述如何调用这些操作的结构细节和每项操作的接口细节，这些直到 WebApp 设计阶段才会给出建议。

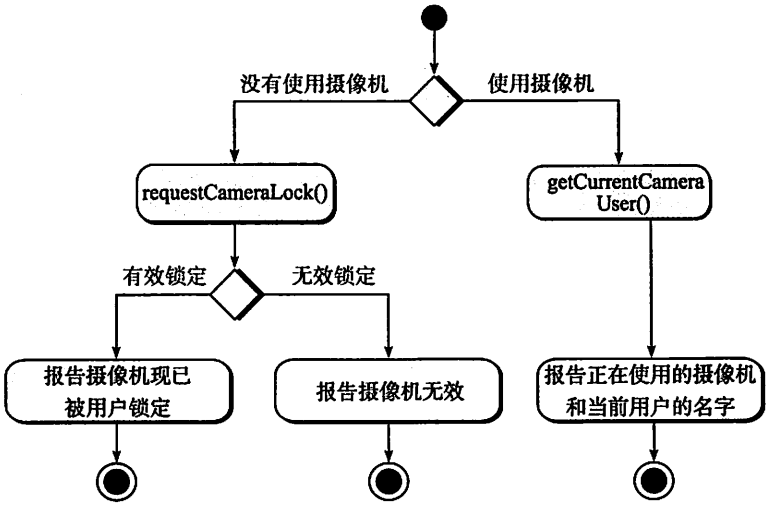

图 11-6　takeControlOfCamera() 操作的活动图

SafeHome WebApp 的扩展功能可能会涉及移动 App 的开发，提供了从智能手机或者平板电脑进入 SafeHome 系统的功能。SafeHome 移动 App 的内容和功能需求可与 WebApp 的子集相似，但还必须建立特殊接口和安全需求。

11.5.7　WebApp 的配置模型

219　在某些情况下，配置模型只不过是服务器端和客户端的属性列表。但是，对更复杂的 App 来说，多种配置的复杂性（例如，多服务器之间的负载分配、高速缓存的体系结构、远程数据库、服务于不同对象的多个服务器）可能对分析和设计产生影响。在必须考虑复杂配置体系结构的情况下，可以使用 UML 部署图。

应该指定通过所有主要的 Web 客户端访问 www.safehomeassured.com 的公共内容和功

能（例如，那些多于 1% 市场占有率的 Web 客户端）。相反，对于少量客户端也可限制复杂的控制和监控功能（只允许 HomeOwner 用户访问）。对移动 App 而言，实施方案可能局限于三种领先的移动操作环境中。www.safehomeassured.com 的配置模型也将与指定的现存产品数据库和监控应用进行交互操作。

11.5.8　导航建模

对于智能手机平台上的大多数移动 App，导航通常局限于相对简单的按键列表和图标菜单。另外，导航的深度（例如超链接的层级数）相对较浅。由于这些原因，导航模型相对简单。

随着 WebApp 和基于平板的移动 App 数量的不断增长，导航建模将更为复杂，常常需要考虑如何让每个用户分类从一个 WebApp 元素（例如内容对象）跳转到另一个元素。我们 220
把导航机制定义为设计的一部分。在这个阶段，分析人员应该关注总体的导航需求，考虑以下问题：

- 某些元素比其他元素更容易到达吗（即需要更少的导航步骤）？表示的优先级别是什么？
- 为了促使用户按他们自己的方向导航，应该强调某些元素吗？
- 应该怎样处理导航错误？
- 导航到相关元素组的优先级应该高于导航到某个特定元素的优先级吗？
- 应该通过链接方式、基于搜索的访问方式还是其他方式来实现导航？
- 根据前面的导航行为，某些确定的元素应该展现给用户吗？
- 应该为用户维护导航日志吗？
- 在用户交互的每一点处，（与单个“返回”链接或有方向的指针相比）完整的导航地图或菜单都可用吗？
- 导航设计应该由大多数普遍期望的用户行为来驱动，还是由已定义的 WebApp 元素可感知的重要性来驱动？
- 为了使将来的使用更加快捷，用户能否“存储”他以前对 WebApp 的导航？
- 应该为哪类用户设计最佳导航？
- 应该如何处理 WebApp 外部的链接？应该覆盖现有的浏览器窗口吗？能否作为一个新的浏览器窗口，还是作为一个单独的框架？

提出并回答这些问题及其他一些问题是导航分析工作的一部分。

软件工程师和其他利益相关者必须决定导航的总体需求。例如，“站点图”会让用户全面纵览整个 WebApp 的结构吗？用户会使用“导游”吗？“导游”将突出显示可以使用的最重要的元素（包括内容对象和功能）吗？用户能够访问内容对象或者由其元素属性所决定的功能吗（例如，用户可能想要访问一个特定建筑的所有图片，或者允许计算重量的所有功能）？

11.6　小结

在需求分析阶段的行为建模描述了软件的动态行为。行为模型采用来自于基于场景、面向流和基于类的输入，从而把分析类和系统的状态作为一个整体来表达。为达到这一目的，要识别状态，定义引起类（或系统）由一个状态转换到另一状态的事件，还要识别完成转换 221
后发生的活动。状态图和顺序图是行为建模的常用表达方式。

分析模式使得软件工程师能够使用现有的知识领域，便于建立需求模型。一个分析模式

描述了一个特定的软件特性或功能，该特性或功能由一套相关用例描述。本章详细说明了模式的目的、使用的动机、使用的限制约束、各种问题域中的可应用性、模式的完整结构、行为和协作以及其他辅助信息。

WebApp 的需求建模能使用本书讨论的大多数的建模元素。但是只能在一套指定的模型中使用这些元素，它涉及 WebApp 中的内容、交互操作、功能、导航和服务客户端的配置。

习题与思考题

11.1 表示行为建模时有两种不同“状态”类型，它们是什么？

11.2 如何从状态图区分顺序图？它们有何相似之处？

11.3 为现代移动手机建议 3 种需求模式，并简约描述每种模式。这些模式能否被其他设备使用？举例说明。

11.4 在问题 11.3 中选择一个你要开发的模式，模拟 11.4.2 节中表达的某个内容和模型，开发出一个合理并完整的模式描述。

11.5 你认为 www.safehomeassured.com 需要多少种分析模型？这些是在 11.5.3 节中描述的每种模型类型所需要的吗？

11.6 WebApp 交互建模的目的是什么？

11.7 引起争议的问题是 WebApp 的功能模型应延期开发直至设计阶段。表述关于这个议题的支持和反对观点。

11.8 配置模型的目的是什么？

11.9 如何从交互模型中区分导航模型？

扩展阅读与信息资源

行为建模呈现了系统行为的动态视图。Samek (《Practical UML Statecharts in C/C++: Event Driven Programming for Embedded Systems》, CRC Press, 2008)、Wagner 和他的同事 (《Modeling Software with Finite State Machines: A Practical Approach》, Auerbach, 2006) 以及 Boerger 和 Staerk (《Abstract State Machines》, Springer, 2003) 深入讨论了状态图和其他行为的表示方法。Gomes
222 和 Fernandez (《Behavioral Modeling for Embedded Systems and Technologies》, Information Science Reference, 2009) 为描述嵌入式系统的行为模型编辑了一本选集。

为软件模式所写的大部分书都关注软件设计。而 Vaughn (《Implementing Domain-Driven Design》, Addison-Wesley, 2013)、Whithall (《Software Requirement Patterns》, Microsoft Press, 2007), Evans (《Domain-Driven Design》, Addison-Wesley, 2003) 和 Fowler ([Fow03], [Fow97]) 的书都是专门写分析模式的。

Pressman 和 Lowe[Pre08] 表述了需要深层次讨论的 WebApp 分析模型。Rossi 和他的同事 (《Web Engineering: Modeling and Implementing Web Applications》, Springer, 2010) 以及 Neil (《Mobile Design Pattern Gallery: UI Patterns》, O'Reilly, 2012) 讨论在应用开发中模式的使用方法。Murugesan 和 Desphande 编写的论文集中包含了一篇文章 (《Web Engineering:Managing Diversity and Complexity of Web Application Development》, Springer, 2001), 处理了 WebApp 需求的各个方面。另外《International Conference on Web Engineering》论文集每年都会发表解决需求建模问题的文章。

网上有大量需求建模方面的资源。关于分析建模的最新参考文献可以在 SEPA 网站 www.mhhe.
223 com/pressman 上找到。