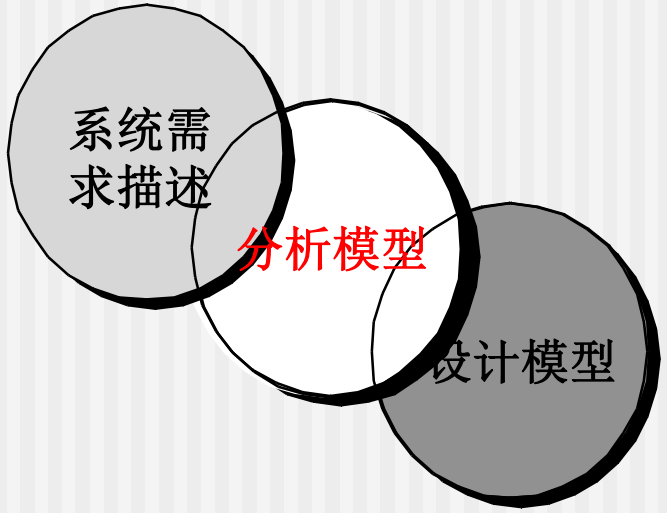
需求分析

- 需求分析让软件工程师（分析师或建模师）细化在前期需求工程的起始、获取、协商任务中获得的基础需求
  - 产生软件工作特征的规格说明
  - 描述软件和其他系统元素的接口
  - 规定软件必须满足的约束

一座桥梁



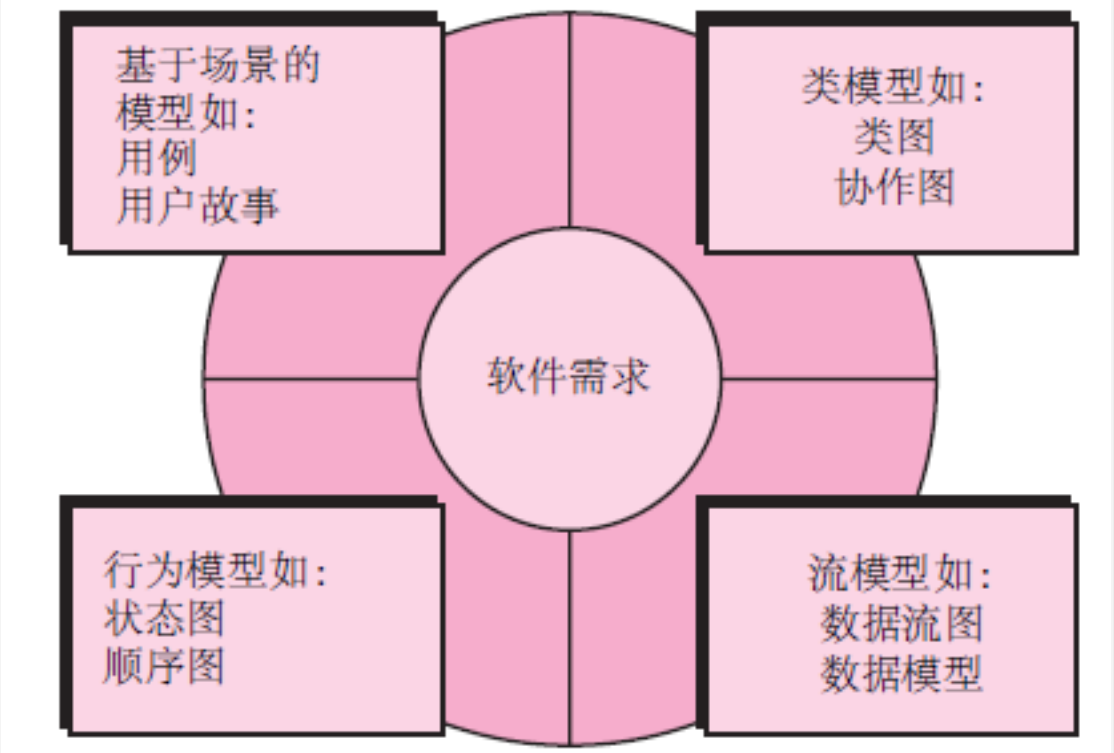
需求分析的原理和目标

- 在需求阶段，软件工程师的主要关注点在做什么，而不是怎么做。是问题定义阶段。
  - 有哪些用户交互？
  - 系统处理什么对象？对象之间有什么关系？
  - 系统必须执行什么功能？（提供什么服务？）
  - 系统展示什么行为？
  - 定义什么接口？
  - 有什么约束？
- 需求模型要实现三个目标：
  - 描述客户需要什么？（问题定义，需求确认）
  - 为软件设计奠定基础？（设计和实现）
  - 定义在软件完成后可以被确认的一组需求？（测试和验收）

需求分析

- 需求分析构建一种或多种模型以描述用户场景、功能活动、类间关系以及当功能元素在系统中运行时怎样进行数据变换

需求建模的元素



需求建模

需求就是对软件期望的行为的表达，需求处理的是对象或实体、它们可能处于的状态，以及用于改变状态或对象特性的功能。需求建模就是对上述的内容的模型化表达。

- 场景模型：来自各种系统“参与者”观点的需求
- 面向类的模型：表示面向对象类（属性和操作）的模型
- 行为或模式模型：描述如何将软件行为表示成外部“事件”的后续响应
- 数据模型：用来描述问题的信息域
- 面向流程的模型：描述系统的功能元素以及它们是如何变换数据的

前面三个模型一般用于OO分析，后面两个是用于结构化分析

分析的经验原则（课本）

- 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。
- 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。
- 关于基础结构和其它非功能的模型应推延到设计阶段再考虑。
- 最小化整个系统内的关联。
- 确认需求模型为所有相关利益者都带来价值。
- 尽可能保持模型简洁。

我的分析经验原则

- 根据收集到的需求识别对象（实体）、对象间的关系，构建基于领域的结构模型。
- 使用恰当的方式表达基于领域的功能模型和行为模型。
- 仅描述系统该实现什么，不描述系统该如何实现（除非用户指定），如特定的算法、数据库选型、编程语言等等。
- 好模型往往是内聚的、最小化互联。
- 尽可能保持模型简洁，深入浅出。

域分析（domain analysis）

- 为了构建和复用分析模式，引入了域分析方法
- 域分析是识别、分析和详细说明某个特定应用领域的共同需求，特别是那些在该应用领域内被多个项目重复使用的……[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。

Donald Firesmith

需求建模的方法

- 结构化分析，一种考虑数据和处理的需求建模方法。（数据流图，ER图）
- 面向对象的分析，关注类的定义和影响客户需求的类之间的协作方式。（UML和统一过程）

需求建模：基于场景的方法

使用UML进行需求建模是从场景建模开始的，即从开发用例、活动图和泳道图形式的场景开始。

用例(场景模型)是用户最容易理解的。

基于场景建模

“用例（use case）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

创建用例要确定:

(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？

编写什么？

- 起始和导出——提供了开始编写用例所需要的信息。
- 运用需求收集会议、质量功能展开(Quality Function Deployment, QFD)和其它需求工程机制需要：
  - 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。

编写多少？

- 随着和利益相关者更深入地交谈，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，可以使用某个结构化的形式重新编写同样的用例。

用例

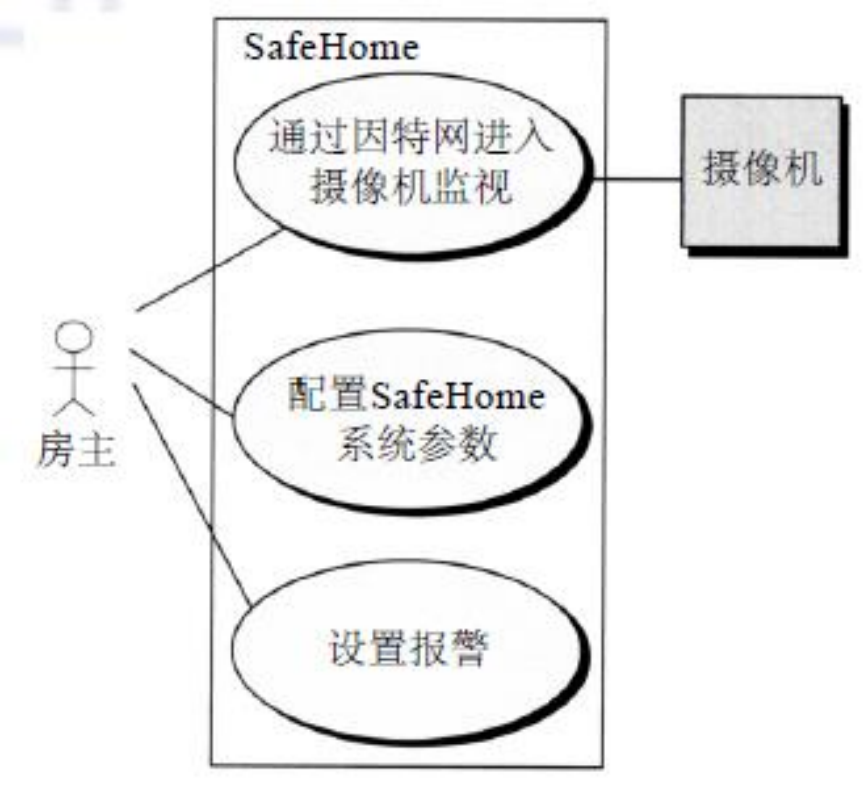
- 一种特定场景描述了系统的“使用线程”（即描述了系统实现的某个业务流程）。
- 参与者（人员或设备）在特定的环境内所扮演的角色
- 用户可在给定场景中扮演多个不同角色

开发用例

- 参与者执行的首要任务或功能是什么？
- 参与者需要获取、生成或改变哪些系统信息？
- 参与者是否需要通知系统外部环境的变化？
- 参与者希望从系统中获取何种信息？
- 参与者是否想了解意料之外的变化？

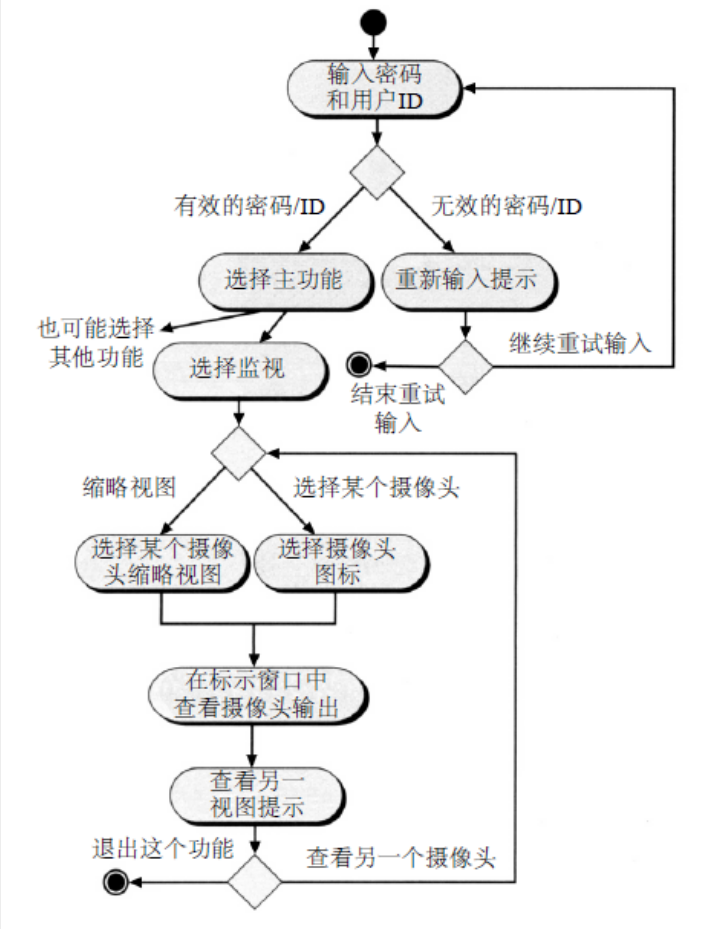
用例一般表示为用例图和用例描述，辅助以活动图

用例图



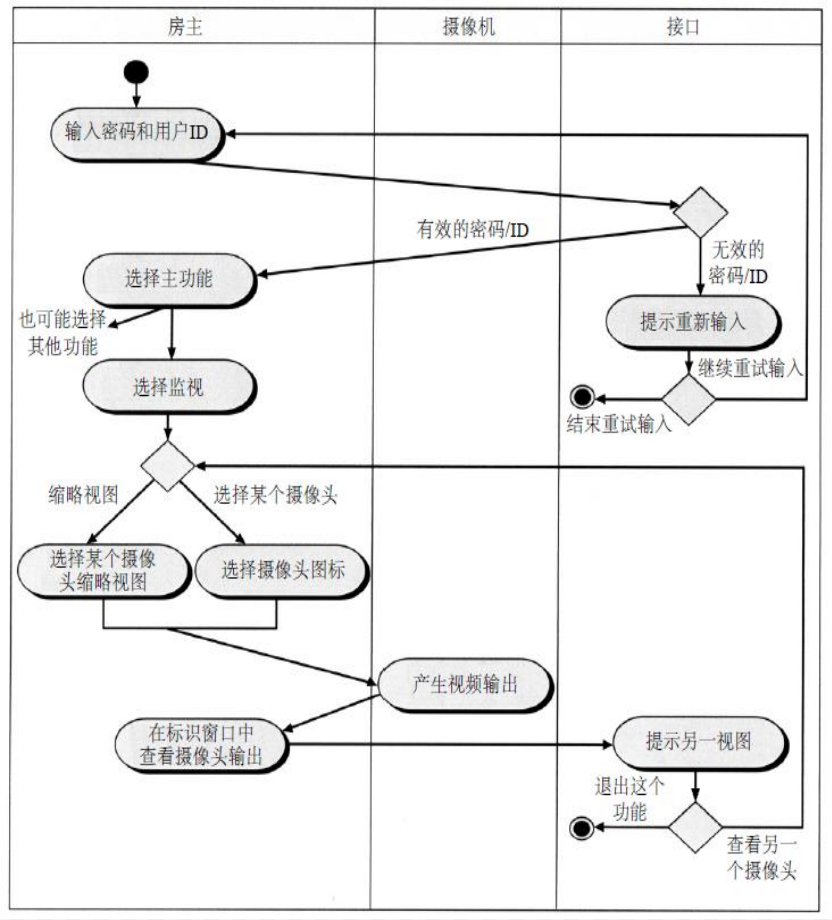
活动图

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



泳道图

UML泳道图是一种用活动图的变形，表示的活动指者该例所描述的同时与行动流的个参执来负责活动



需求建模：基于类的方法

软件问题总是以一套交互对象为特征的。

基于类的建模定义了对象、属性和关系。

基于类建模

- 类模型定义了：
  - 系统操作的对象
  - 应用于对象的操作（也称为方法或服务）
  - 对象间的关系（某种层级）
  - 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
  - 属性和操作
  - 类的职责协作者(CRC)模型
  - 协作图
  - 包

识别分析类

- 正确地识别类是类建模中很困难的一件事情。
- 通过检查需求模型开发的需求用例，对系统开发的用例进行“语法解析”[Abb83]
  - 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？

分析类的表现方式

- 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、信件、信号），问题信息域的一部分。
- 偶发事件或事件（例如，所有权发生转移、或完成了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），建立问题和系统整体功能的上下文环境。
- 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。

分析类的划分方法不是唯一的

示例

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。

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……

示例

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		

类模型类应满足的特征

- 保留信息。只有记录该潜在类的信息才能保证系统正常工作，它在分析过程中才是有用的。
- 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。
- 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。
- 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。
- 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。
- 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。

示例

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

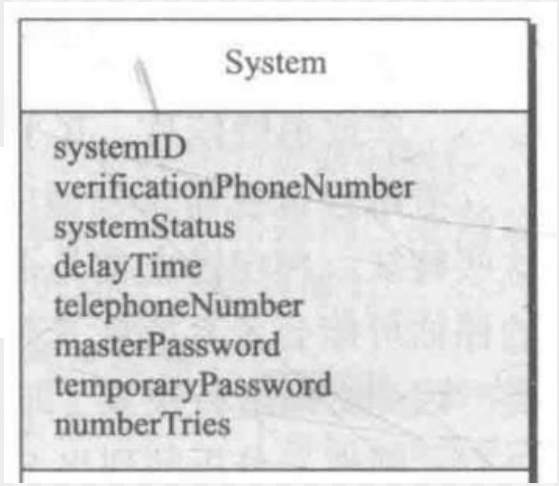
描述属性

- 属性用来描述需求模型选择的类，属性定义了类。
- 类的属性选择往往与问题空间密切相关。
  - 为职业棒球手建立两种不同的类
    - **数据统计系统：**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
    - **养老系统：**属性是与平均工资、充分享受优惠权后的信用、所选的养老计划、邮件地址等相关的。

描述属性

- 什么数据项（组合项或者基本项）能够在当前上下文中完整地表征这个类？
- 不要把其他类定义为属性
- SafeHome中的System类

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



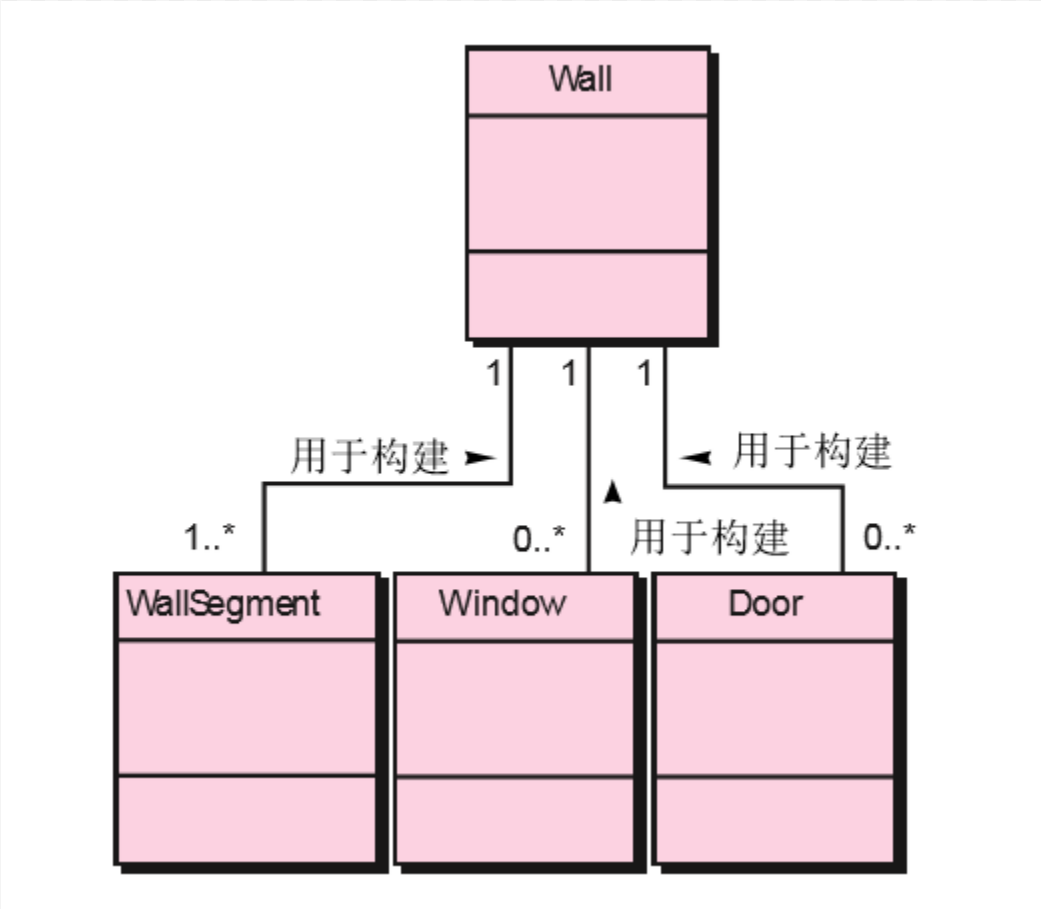
定义操作

- 操作定义了某个对象的行为
- 操作通常可以划分为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）事件监听操作，监视某个对象发生某个控制事件。

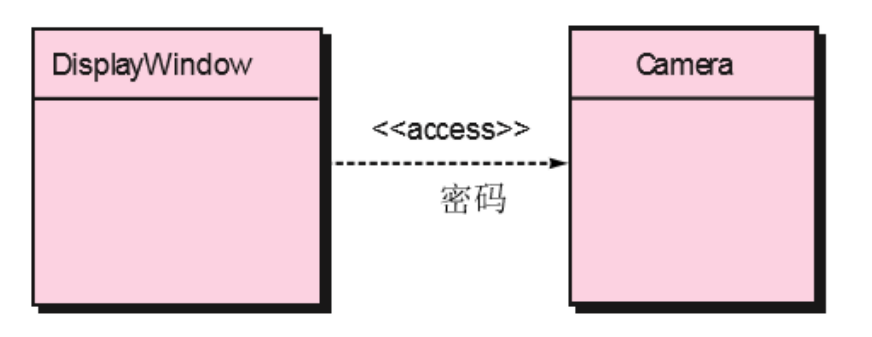
关系和依赖

- 分析类之间往往以某种方式相互联系着。
  - 在UML中，这些联系被称作关联（accociations）
  - 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。

多样性



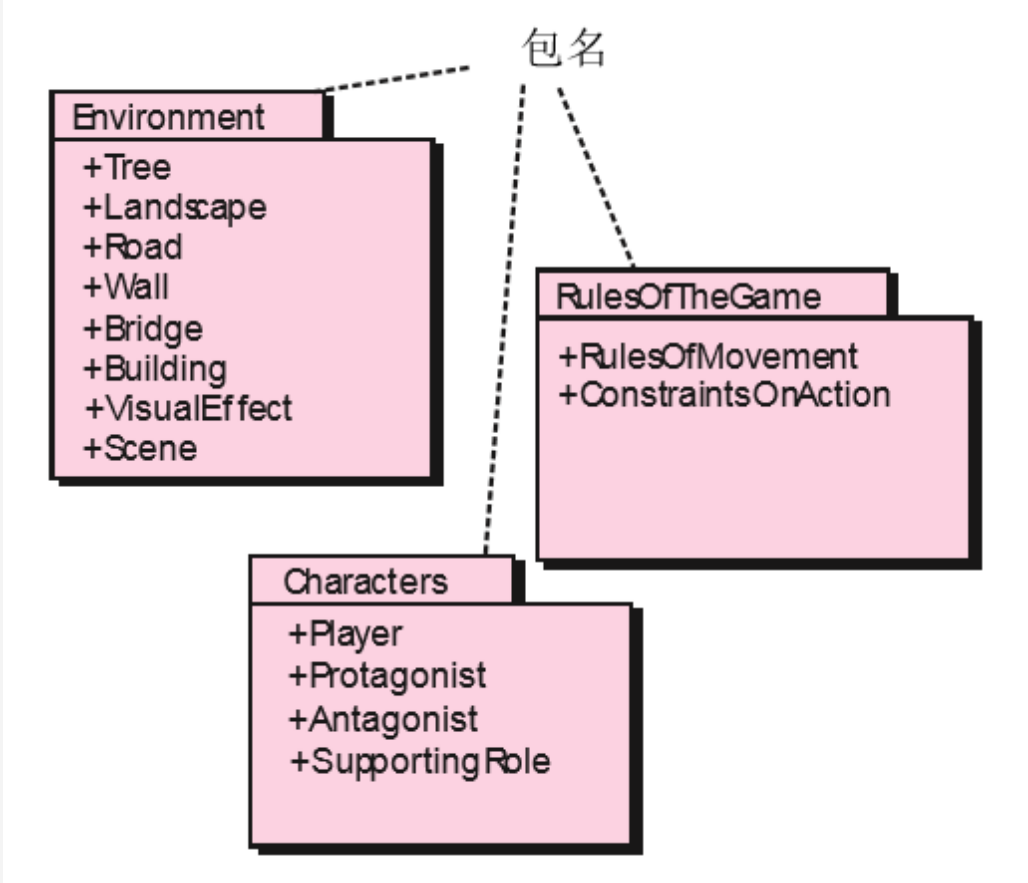
依赖



分析包

- 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。
- 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。

分析包


包名
Environment
+Tree
+Landscape
+Road
+Wall
+Bridge
+Building
+VisualEffect
+Scene
RulesOfTheGame
+RulesOfMovement
+ConstraintsOnAction
Characters
+Player
+Protagonist
+Antagonist
+SupportingRole

CRC模型

- 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。

CRC建模


Class: 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
显示摄像头的位置
Wall
Camera

类的分类

■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）。

■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）。

■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

- 实体类的创建或更新；
- 当边界类从实体对象获取信息后的实例化；
- 对象集合间的复杂通信；
- 对象间或用户和应用系统间交换数据的确认。

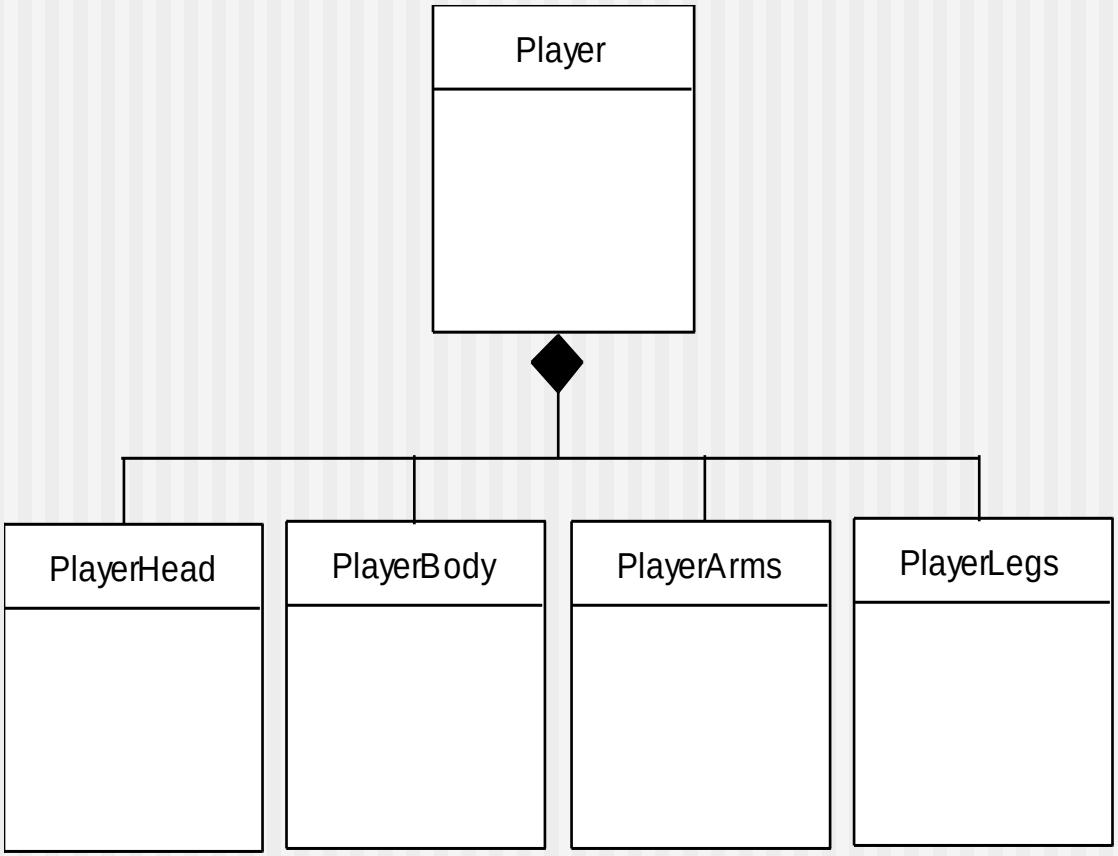
职责

- 智能系统应分布在所有类中以求最佳地满足问题的需求。
- 每个职责的说明应尽可能具有普遍性。
- 信息和与之相关的行为应放在同一个类中。
- 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。
- 适合时，职责应由相关类共享。

协作

- 类有一种或两种方法实现其职责：
  - 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；
  - 一个类可以和其他类协作。
- 协作定义了类之间的关系
- 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。
- 类之间三种不同的通用关系[Wir90]：
  - *is-part-of*（是……一部分）关系
  - *has-knowledge-of*（有……的知识）关系
  - *depends-upon*（依赖……）关系

复合聚合类



评审CRC模型

- 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）。
- 分类管理所有的用例场景（以及相关的用例图）。
- 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。
- 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。
- 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。

需求建模：行为

行为建模（包括模式）通常是用例和类模型的补充

行为模型描述了系统的动态特征

生成行为模型

- 行为模型显示了软件如何对外部事件或激励作出响应。要生成模型，分析师必须按如下步骤进行：
  - 评估所有的用例，以保证完全理解系统内的交互顺序。
  - 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。
  - 为每个用例生成序列。
  - 创建系统状态图。
  - 评审行为模型以验证准确性和一致性。

识别用例事件

- 参与者和系统之间交换了信息就发生了事件。
- 事件不应该是被交换的信息，而是已交换信息的事实。

例如：

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。

状态表达

- 在行为建模的场合下，必须考虑两种不同的状态描述：
  - 系统执行其功能时每个类的状态
  - 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。
  - 一个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。
- 两种行为的表现形式：分析类的状态图和顺序图。

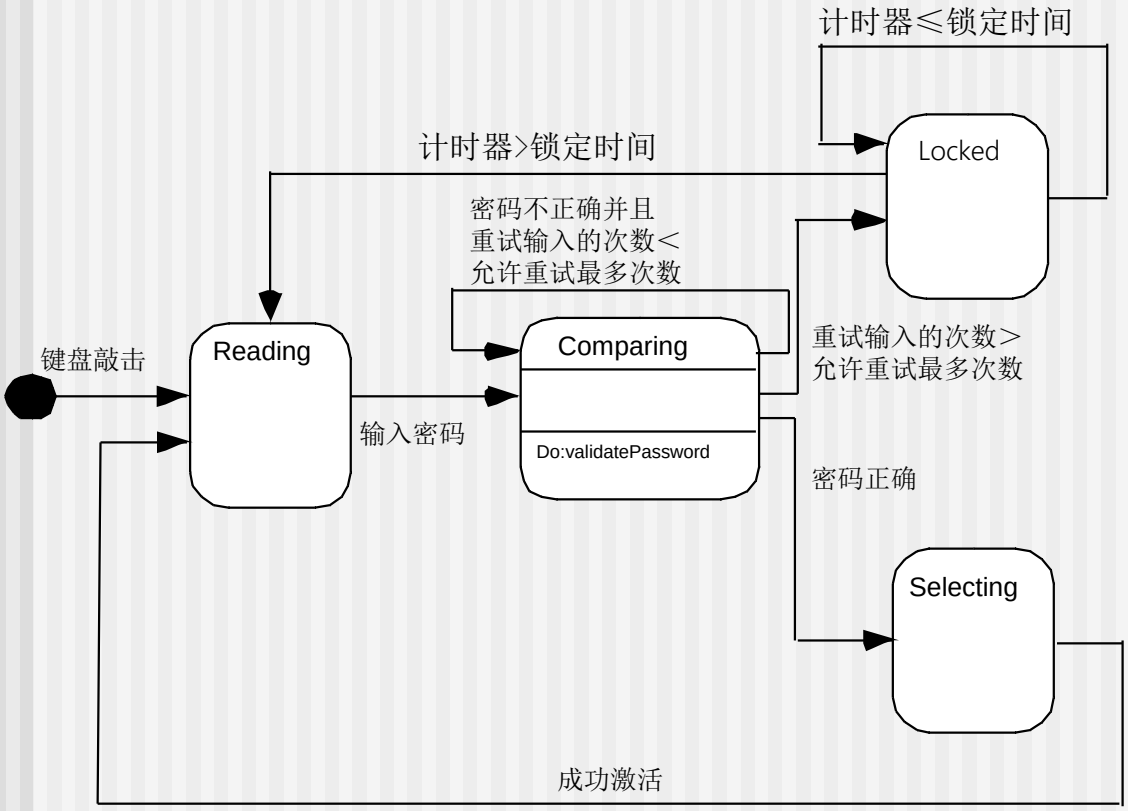
行为建模

- 列出系统的不同状态（系统如何表现？）
- 指出系统如何从一种状态转移到另一种状态（系统如何改变状态？）
  - 指出事件
  - 指出动作
- 绘制状态图或顺序图

系统状态

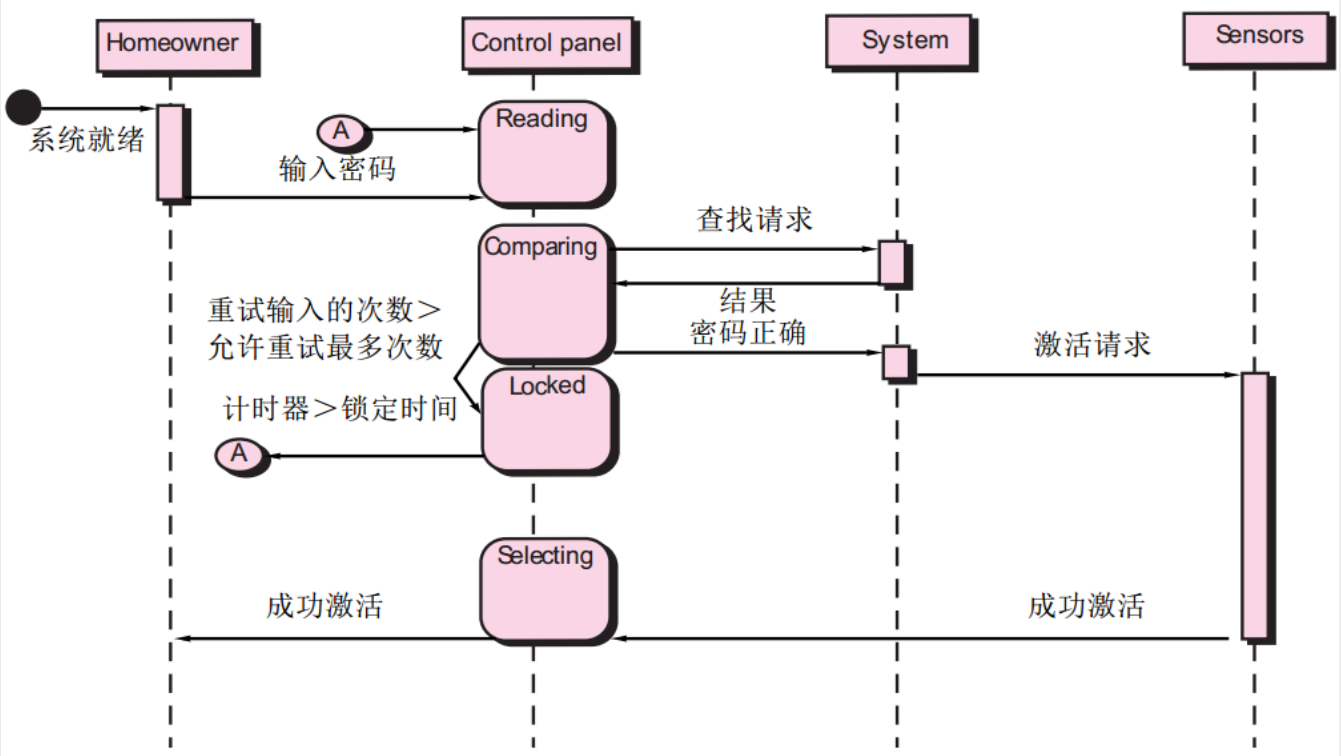
- 状态：一组可观察到的情况下，在给定的时间描述系统的行为
- 状态转移：从一个状态转移到另一个状态
- 事件：发生时系统将表现出某种可以测的行为
- 动作：与状态转移同时发生的或者它作为状态转移的结果

ControlPanel类的状态图



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，箭头上的标注是触发状态转移的事件

顺序图



描述事件是如何引发从一个对象到另一个对象的转移

需求建模的模式

- 软件模式是获取领域知识的一种机制从而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，则进行记载。

这部分内容请同学们自学

发现分析模式

- 在需求模型的描述中最基本的元素是用例。
- 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）“是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]。

示例

- 考虑要求控制和监控“实时查看摄像机”和汽车临近传感器的软件用例：

用例：监控反向运动

描述：当车辆安装了反向齿轮，控制软件就能从后向视频摄像机将一段视频输入到仪表板显示器上。控制软件在仪表板显示器上叠加各种各样距离和方向的线，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3英尺内就会让车自动停止。

示例

- 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和在一个物理系统中对传感器和执行器的控制。
- 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——**Actuator-Sensor**（执行器–传感器）。

执行器-传感器 模式—I

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。

动机：嵌入系统常有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十分不同，但它们的行为是相似的，足以让它们构成一个模式。这个模式显示了如何为一个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有某种方法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。

每个执行者必须有某种方法调用由**ComputingComponent**计算构件决定的适当应答。

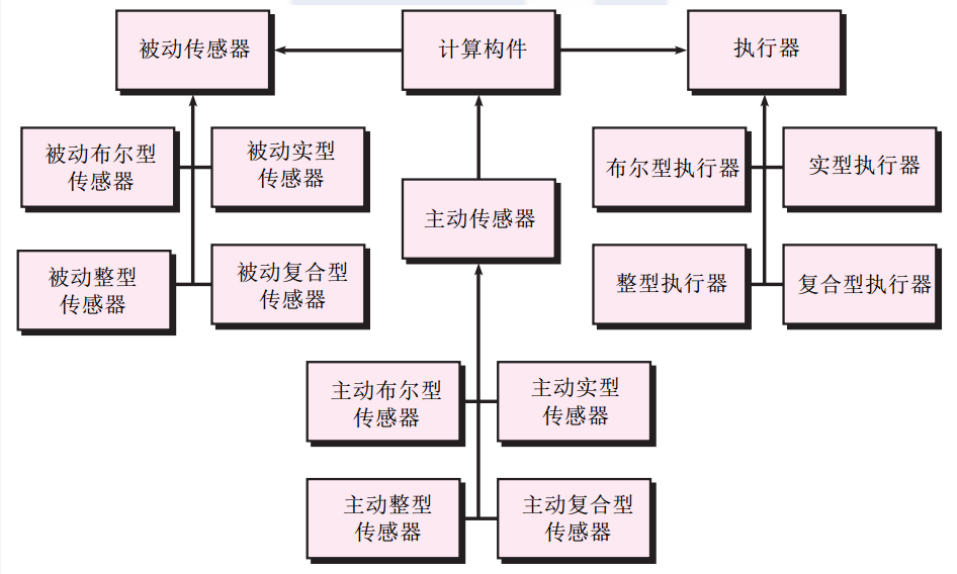
每个传感器和执行器应有实施检测其自身操作状态的功能。

每个传感器和执行器能测试接受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。

执行器-传感器 模式—II

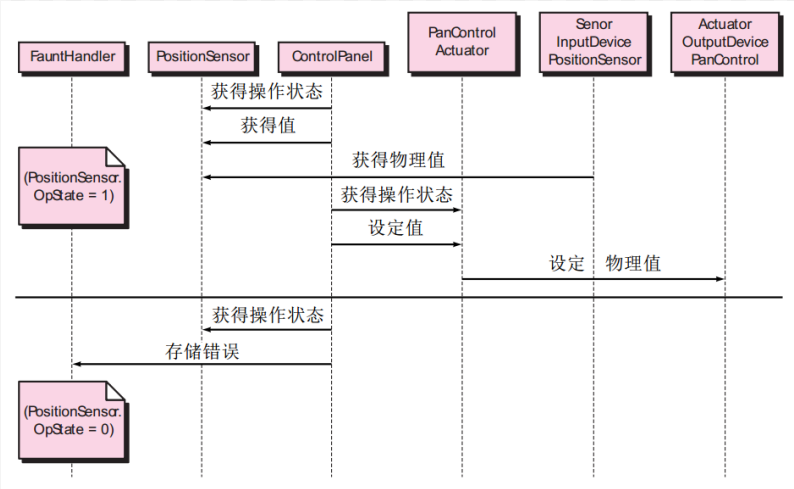
适用性：对有多个传感器和执行器的任何系统都是非常有用的。

结构体：图7.8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。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。虽然如此但是这些设备还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。



执行器-传感器 模式—III

行为：图7.9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、聚焦）。这里在读取或设置一个值时，**ControlPanel**需要一个传感器（被动传感器）和一个执行器（摇摄控制器）进行以诊断为目的的核查操作状态。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，它将决定这些错误如何影响系统，并需要采取什么措施。从传感器获得的数据经过计算得到执行器所需的应答。



Web应用系统的需求建模

内容分析。给出由应用系统提供的全部系列内容。内容包括文本、图表和图像、视频和音频数据。数据建模可以用来识别和描述每个数据对象。

交互分析。描述了用户与应用系统采用了哪种交互方式。可开发用例更加具体的描述这种交互。

功能分析。 作为交互分析的一部分，创建使用场景（用例）定义了将用于处理应用系统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却是最终用户所必需的。详细描述所有操作和功能。

配置分析。详细描述应用系统所在的环境和基础设施。

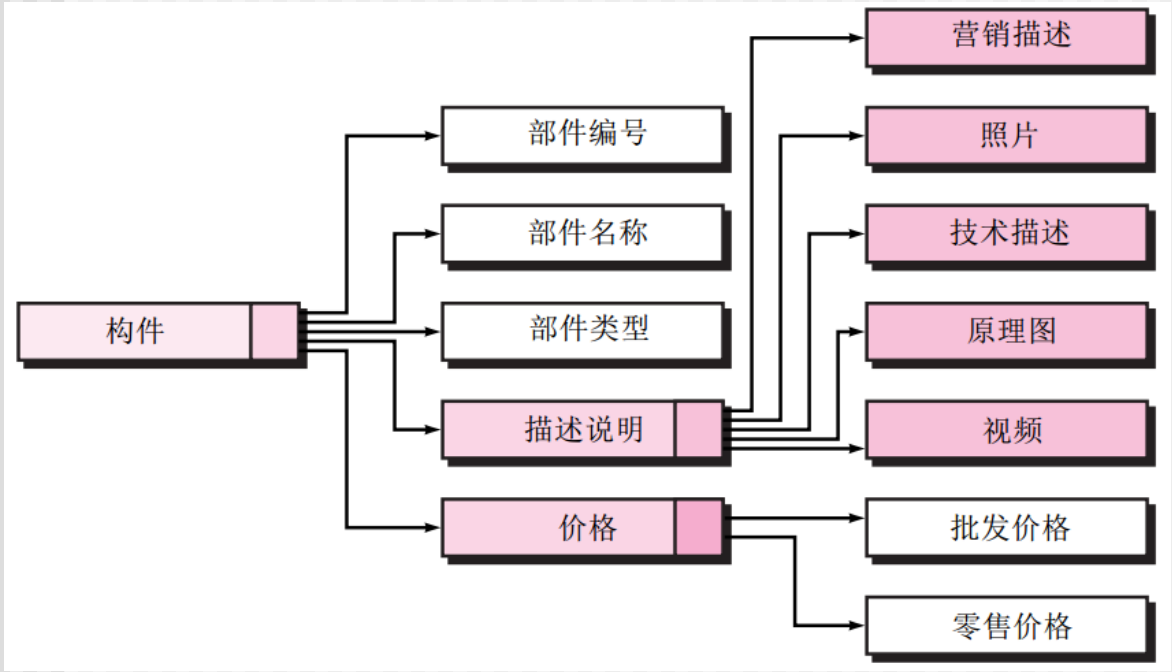
何时进行分析？

- 在某些Web或手机App的应用场景下，分析与设计步骤合并。然而，当出现下列情形时，需要进行更加详尽的分析：
  - 应用的规模和复杂度的增长
  - 利益相关者的人数庞大
  - 开发者的人数庞大
  - 一起工作以前该Web应用团队成员的级别
  - 组织成功的程度直接依赖于Web应用的成功

内容建模

- 内容对象是从用例中提取的
  - 检查直接或间接引用内容的场景描述
- 每个内容对象的属性都是可识别的
- 内容对象之间的关系和/或内容的层次结构由WebApp维护
  - 关系——实体-关系图或UML
  - 层次——数据树或UML

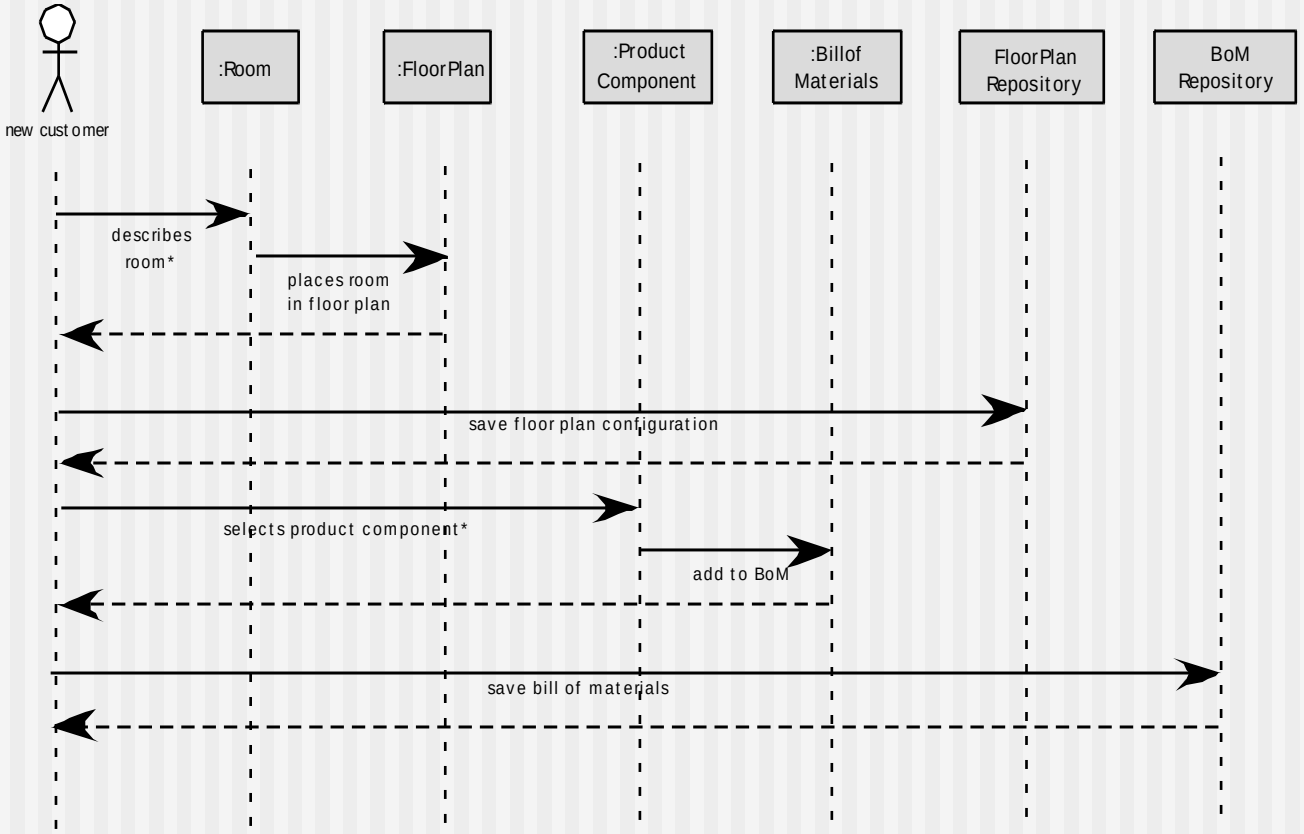
数据树



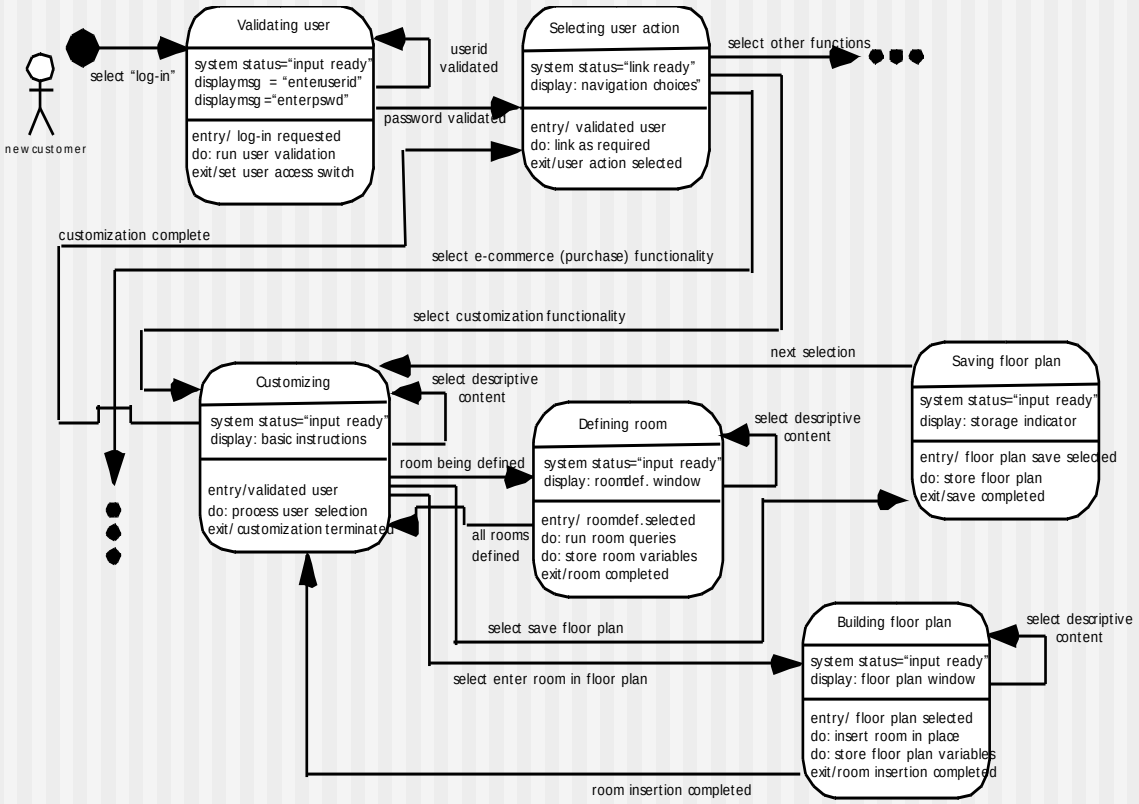
交互模型

- 由4种元素组成：
  - 用例
  - 顺序图
  - 状态图
  - 用户界面原型
- 每种元素都是重要的UML符号，具体将在附录中详述

顺序图



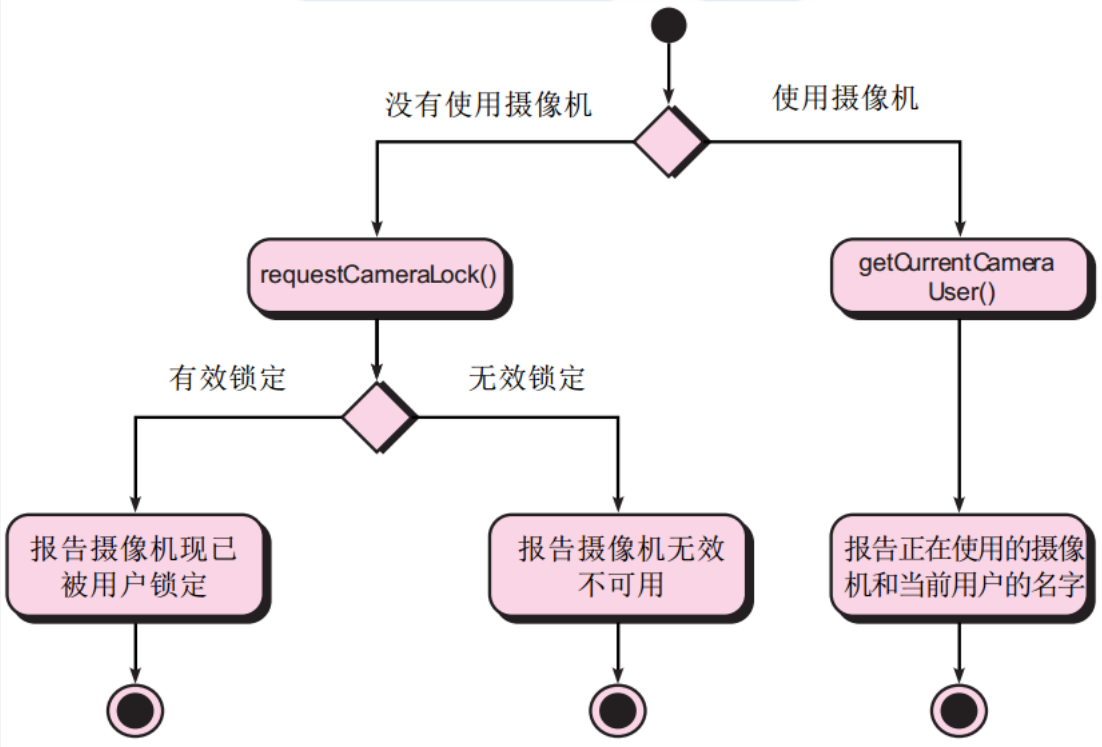
状态图



功能模型

- 功能模型描述Web应用系统的两个处理元素
  - 用户可观察到的功能是由Web应用系统传递给最终用户的；
  - 分析类中的操作实现与类相关的行为。
- UML的活动图可用来表示处理细节

活动图



配置模型

- 服务器端
  - 必须指定服务器硬件和操作系统环境
  - 必须考虑服务器端的互操作性
  - 必须指定合适的接口、通信协议和相关合作信息
- 客户端
  - 浏览器的配置文件必须可识别
  - 应该定义测试需求

导航模型 -I

- 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？
- 为了促使用户以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权应该高于导航到某个特定元素的优先权吗？
- 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？

导航模型 -II

- 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）一个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Web应用系统元素可感知的重要性来驱动？
- 为了促进将来使用快捷，一个用户能否“存储”他以前对Web应用系统的导航？
- 应该为哪类用户设计最佳导航？
- 应该如何处理Web应用系统外部的链接？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？