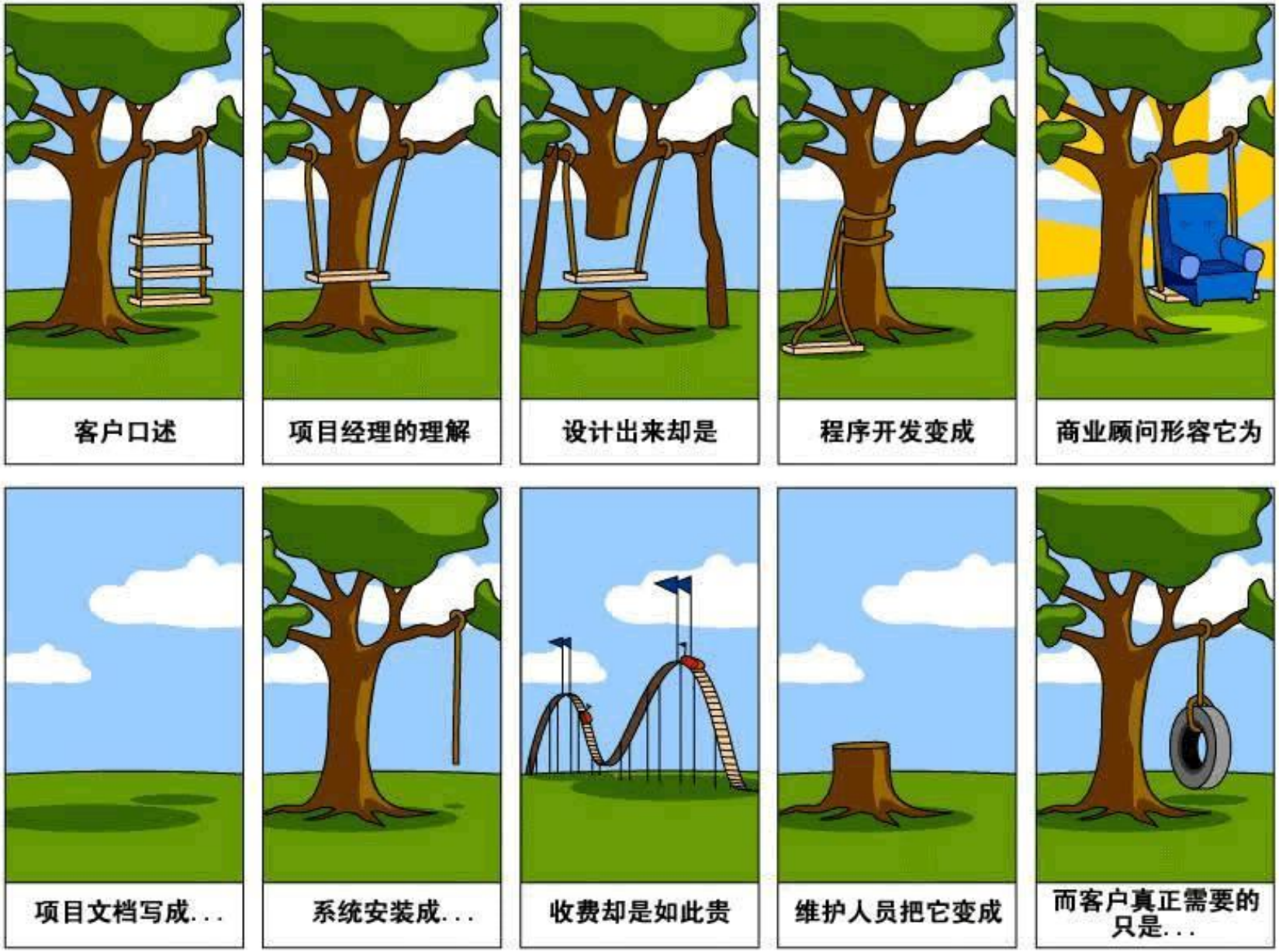
软件工程

第7章 理解需求

徐本柱 软件学院

2018-09


客户口述
项目经理的理解
设计出来却是
程序开发变成
商业顾问形容它为
项目文档写成...
系统安装成...
收费却是如此贵
维护人员把它变成
而客户真正需要的只是...

主要内容

- 需求工程
- 建立根基
- 获取需求
- 开发用例
- 构建分析模型
- 避免常见错误

要点浏览

- 需求工程帮助软件工程师更好地理解他们将要解决的问题
- 包含的一系列任务有助于理解软件
  - 如何影响业务
  - 客户想要什么
  - 最终用户将如何和软件交互。
- 软件工程师和项目利益相关者都参与需求工程

要点浏览

- 需求工程的重要性
  - 在设计和开发某个优秀的计算机软件时
  - 如果软件解决的问题是错误的，即使软件再精巧也满足不了任何人的要求
  - 设计和开发一个基于计算机的系统之前，理解客户需求非常重要
- 步骤：
  - 起始（定义问题的范围和性质）
  - 获取（定义需要什么）
  - 细化（精确定义、修改基本需求）
  - 明确和评审
- 工作产品和质量保证措施

面临的问题

- 理解需求—最困难的任务之一
  - 客户难道不知道需要什么？
  - 最终用户难道对给他们带来实际收益的特征和功能没有清楚的认识？
  - 很多情况下的确是这样的
  - 即使用户和最终用户清楚地知道他们的要求，这些要求也会在项目的实施过程中改变

7.1 需求工程

- 需求工程—致力于不断理解需求的大量任务和技术
  - 从软件过程的角度来看，需求工程是一个软件工程动作
  - 始于沟通并持续至建模
- 需求工程在设计和构造之间建立起联系的桥梁
  - 有人认为源于项目利益相关者，即在那里定义业务需求，刻画用户场景，描述功能和特性，识别项目约束条件。
  - 其他人可能会建议从宽泛的系统定义开始，此时软件只是更大的系统范围中的一个构件。
  - 不管起始点在哪里，横跨这个桥梁将到项目之上更高的层次：
    - 软件团队检查将要进行的软件工作的内容
    - 必须提交设计和构建需要的特定要求
    - 完成指导工作顺序的优先级定义
    - 以及将深切影响随后设计的信息、功能和行为。

需求工程任务

- 需求工程为以下工作提供了良好的机制：
  - 理解客户需要什么、分析要求、评估可行性、协商合理的方案、无歧义地详细说明方案、确认规格说明、管理需求以至将这些需求转化为可运行系统。
- 需求工程包括七项明确的任务：
  - 起始、获取、细化、协商、规格说明、确认和管理。

起始

- 项目通常都是在确定了商业要求或发现了潜在的新市场、新服务时才开始
  - 业务领域的利益相关者定义业务用例，确定市场的宽度和深度，进行粗略的可行性分析，并确定项目范围的工作说明
- 建立基本的理解，包括
  - 存在的问题
  - 谁需要解决方案
  - 解决方案的性质
  - 与项目利益相关者和开发人员之间达成初步交流合作的效果

获取

- 征求各利益相关者的需求，回答问题：
  - 系统或产品的目标是什么？
  - 想要实现什么？
  - 系统和产品如何满足业务的要求？
  - 最终系统或产品如何用于日常工作？
- 实际上获取需求非常困难。

获取

- 范围问题：系统的边界不清楚，或客户/用户的说明带有多余的技术细节，这些细节可能会混淆而不是澄清系统的整体目标。
- 理解问题：客户/用户并不完全确定需要什么
  - 对其计算环境的能力和限制所知甚少
  - 对问题域没有完整的认识
  - 与系统工程师在沟通上有问题
  - 忽略略那些他们认为是“明显的”信息
  - 确定的需求和其他客户/用户的需求相冲突
  - 需求说明有歧义或不可测试。
- 易变问题：需求随时间变化。

细化

- 开发一个需求模型，来说明软件的功能、特征和信息的各个方面
  - 由一系列的用户场景建模和求精任务驱动
  - 每个用户场景被分解为精炼分析类（业务域实体）
  - 定义每个分析类的属性
  - 确定每个类所需要的服务
  - 确定类之间的关联和协作关系
  - 完成各种UML图作为补充。
- 精化的最终结果
  - 形成一个精确的需求模型
  - 以说明软件的功能、特征和信息的各个方面

协商

- 形成能令开发人员和客户都满意的可交付系统
  - 业务资源有限，客户提出的过高的要求
  - 不同客户提出相互冲突的需求
- 需求工程师通过协商过程来调解这些冲突
  - 让客户、用户和利益相关者对各自的需求排序
  - 然后按优先级讨论冲突
- 使用迭代的方法
  - 需求排序，评估每项需求的成本和风险
  - 处理内部冲突，删除、组合或修改需求
  - 以便参与各方均能达到一定的满意度

规格说明

- 规格说明可以是下面的一个（或者多个）：
  - 一份写好的文档、
  - 一套图形化的模型、
  - 一个形式化的数学模型，
  - 一组使用场景、
  - 一个原型。
- 在开发规格说明时保持灵活性有时是必要的
  - 大型系统最好采用自然语言描述和图形化模型来编写
  - 技术环节明确的较小产品或系统，使用场景即可满足

确认

- 确认：对需求工程的工作产品进行质量评估
- 需求确认要检查规格说明以保证：
  - 所有的系统需求已被无歧义地说明
  - 不一致性、疏漏和错误已被检测出并被纠正
  - 工作产品符合为过程、项目和产品建立的标准
- 正式技术评审是最主要的需求确认机制
- 一组检查机制来发现
  - 查找内容或解释上的错误
  - 需要进一步解释澄清的地方、丢失的信息
  - 不一致性(建造大型产品或系统时遇到的主要问题)
  - 冲突的需求或不现实的（不能达到的）需求

需求确认检查单

- 需求说明清晰吗？有没有可能造成误解？
- 需求的来源（如人员、规则、文档）弄清了吗？需求的最终说明是否已经根据或对照最初来源检查过？
- 需求是否用定量的术语界定？
- 其他哪些需求与此需求相关？是否已经使用交叉索引或其他机制清楚地加以说明了？
- 需求是否违背某个系统领域的约束？
- 需求是否可以测试？如果可以，能否说明测试检验了需求？
- 对已经创建的任何系统模型，需求是否可跟踪？
- 对整体系统/产品目标，需求是否可跟踪？
- 规格说明的构造方式是否有助于理解、引用和翻译成更技术性的工作产品？
- 对已创建的规格说明是否建立了索引？
- 和系统性能、行为及运行特征相关的需求的说明是否清楚？哪些需求是隐含出现的？

需求管理

- 基于计算机的系统其需求会变更
- 且变更的要求贯穿于系统的整个生存期。
- 需求管理是用于帮助项目组在项目进展中
  - 标识、控制和跟踪需求
  - 以及需求变更的一组活动。
- 大部分和软件配置管理（SCM）技术相同

7.2 建立根基

- 在理想情况下，利益相关者和软件工程师在同一个小组中工作。在这种情况下，需求工程就只是和组里熟悉的同事进行有意义的交谈。
- **但实际情况往往不是这样**
- 客户可能正在不同的城市或国家，对于想要什么可能仅有模糊的想法，对于将要构建的系统可能存在不同的意见，技术知识可能很有限，可能仅有有限的时间与需求工程师沟通。
- **软件开发团队经常被迫在这种环境的限制下工作**
- 启动需求工程所需步骤：

确认利益相关者

- 利益相关者—
  “直接或间接从正在开发的系统中获益的人”
- 可以确定如下几个容易理解的利益相关者：
  - 业务运行管理人员、产品管理人员、市场营销人员、内部和外部客户、最终用户、顾问、产品工程师、软件工程师、支持和维护工程师以及其他人员
- 每个利益相关者对系统都有不同的考虑，
  - 当系统成功开发后所能获得的利益也不相同
  - 当系统开发失败时所面临的风险也不同
- **“你认为我还应该和谁交谈？”**

识别多种观点

- 系统需求调研从不同的视角展开（利益相关者）
- 参与者中的每个人都将为需求工程贡献信息
- 多个角度收集，可能存在不一致性或矛盾的需求
- 需求工程师对利益相关者提供的信息分类
  - 分类方法便于决策者为系统选择内部一致的需求集合

协同合作

- 需求工程师的工作是
  - 标识公共区域（即所有利益相关者都同意的需求）和
  - 矛盾区域或不一致区域（即某个利益相关者提出的需求和其他利益相关者的需求相矛盾）
- 矛盾区域的解决
  - 使用优先点

基于“优先点”的“投票”方案

- 所有的利益相关者都会分配到一定数量的优先点，这些优先点可以适用于很多需求
- 在需求列表上，每个利益相关者通过向每个需求分配一个或多个优先点来表明该需求的相对重要性（个人观点）
- 优先点用过之后就不能再次使用，一旦某个利益相关者的优先点用完，他就不能再对需求实施进一步的操作
- 所有利益相关者在每项需求上的优先点总数显示了该需求的综合重要性。

首次提问

- 第一组与环境无关的问题集中于客户和其他利益相关者、整体目标、收益。
  - 谁是这项工作的最初提出者？
  - 谁将使用该解决方案？
  - 成功的解决方案将带来什么样的经济收益？
  - 对于这个解决方案你还需要其他资源吗？

首次提问

- 下一组问题有助于软件开发组更好地理解问题，并允许客户表达其对解决方案的看法。
  - 如何描述由某成功的解决方案产生的“良好的”输出？
  - 该解决方案强调了什么问题？
  - 能展示（或描述）解决方案的使用环境吗？
  - 存在影响解决方案的特殊性能问题或约束？

首次提问

- 最后一组问题关注于沟通活动本身的效率。
  - 你是回答这些问题的合适人选吗？
  - 你的回答是“正式的”吗？
  - 我的提问和你想解决的问题相关吗？
  - 我的问题是否太多了？
  - 还有其他人员可以提供更多的信息吗？
  - 还有我应该问的其他问题吗？

首次提问

- 这些问题
  - 有助于“打破坚冰”
  - 有助于交流的开始
  - 对成功获取需求至关重要
- 会议形式的问与答不一定是取得成功的好方法
  - 事实上，Q&A会议应该仅仅用于首次接触
  - 然后用问题求解、协商和规格说明等方式来取代

7.3 获取需求

- 需求获取（需求收集）
  - 将问题求解、精化、谈判和规格说明等方面的元素
  - 结合在一起
- 利益相关者和开发人员共同完成如下任务：
  - 确认问题
  - 为解决方案的要素提供建议，
  - 商讨不同的方法
  - 描述初步的需求解决方案。

7.3.1 协作收集需求

- 会议由软件工程师和其他的共利益者共同举办和参与
- 制定筹备和参与会议的规则
- 拟定一个会议议程
- 由一个“主持人”（可以是客户、开发人员或其他人）控制会议
- 采用“方案论证手段”（可以是工作表、活动挂图、不干胶贴纸或电子公告牌、聊天室或虚拟论坛）
- 目标是
  - 标识问题
  - 提出解决方案的要素
  - 协商不同方法
  - 确定一套解决需求问题的初级方案

协作收集需求

- 在需求的起始阶段，基本问题和问题的答案确定了问题的范围和对解决方案的整体理解
- 除了这些最初的会议之外，利益相关者要写一个1~2页的“产品要求”。
- 选择会议地点、时间和日期，选举主持人
- 来自开发组和其他利益相关者组织的代表被邀请出席会议，在会议召开之前应将产品要求分发给所有的与会者。

SafeHome实例[1]

我们的研究表明，住宅管理系统市场以每年40%的速度增长。我们推向市场的首个SafeHome功能将是住宅安全功能，因为多数人都熟悉“报警系统”，所以这将更容易销售。

住宅安全功能应该为各种不希望出现的“情况”提供保护，如非法入侵、火灾、漏水、一氧化碳浓度超标等等。该功能将使用无线传感器监控每种情况，户主可以编程控制，并且在发现情况时自动电话联系监控部门。

协作收集需求

- 在召开会议评审产品要求的前几天，要求每个与会者列出构成系统周围环境的对象、由系统产生的其他对象以及系统用来完成功能的对象。
- 另要求每个与会者列出服务操作或与对象交互的服务（过程或功能）列表。
- 还要开发约束列表（如成本、规模大小、业务规则）和性能标准（如速度、精确度）。
- 这些列表不要求完美无缺，但要反映每个人对系统的理解。

SafeHome实例[2]

每个人都要给出上面所说的列表。SafeHome描述的对象可能包括：一个控制面板、若干烟感器、若干门窗传感器、若干动态检测器、一个警报器、一个事件（一个已被激活的传感器）、一个显示器、一台计算机、若干电话号码、一个电话等。服务列表可能包括：配置系统、设置警报器、监测传感器、电话拨号、控制面板编程以及读显示器（注意，服务作用于对象）。采用类似的方法，每个与会者都将开发约束列表（例如，当传感器不工作时系统必须能够识别，必须是用户友好的，必须能够和标准电话线直接连接）和性能标准列表（例如，一个传感器事件应在一秒内被识别，应实施事件优先级方案）。

协作收集需求

- 这些对象列表可以用大的纸张钉在房间的墙上或用便签纸贴在墙上或写在墙板上
- 列表也可以贴在内部网站的电子公告牌上或聊天室中，便于会议前的评审。
- 理想情况下，应该能够分别操作每个列表项，以便于合并列表、删除项以及加入新项。
- 在本阶段，严禁批评和争论。

协作收集需求

- 当某个话题域的各个列表被提出后，小组将生成一个组合列表
- 该组合列表将删除冗余项，并加入在讨论过程中出现的一些新的想法，但是不删除任何东西
- 在所有话题域的组合列表都生成后，主持人开始讨论协调
  - 组合列表可能会缩短、加长或重新措词，以求更恰当地反映即将开发的产品/系统
  - 目标是为每个话题域（对象、服务、约束或性能）提交一个意见一致的列表，在后面的工作中将用到这个列表。

协作收集需求

- 一旦完成意见一致的列表，
- 团队被分为更小的子团队，
- 每个子团队试图为每个列表中的一个或多个项目编写小规格说明。
- 每个小规格说明是对包含在列表中的单词或短语的精炼。

SafeHome实例[3]

- **SafeHome**对象控制面板的小规格说明：控制面板是一个安装在墙上的装置，尺寸大概是9×5英寸；控制面板和传感器、计算机之间是无线连接；通过一个12键的键盘与用户交互，通过一个3×3的LCD彩色显示器为用户提供反馈信息；软件将提供交互提示、回显以及类似的功能。

协作收集需求

- 每个子团队将他们完成的每个小规格说明提交给所有的与会者讨论，进行添加、删除和进一步细化等工作。
- 在某些情况下，编写小规格说明可能会发现新的对象、服务、约束或性能需求，可以将这些新发现加入到原始列表中。
- 在所有的讨论过程中，团队可能会提出某些在会议上不能解决的问题，将这些问题列表保留起来以便这些意见在以后的工作中发挥作用。

7.3.2 质量功能部署

- 质量功能部署（ QFD ）
  - 一种将客户要求转化成软件技术需求的技术
  - QFD强调理解“什么是对客户有价值的”，
  - 然后在整个工程活动中部署这些价值。

QFD确认的需求

- 常规需求：会议中向客户陈述一个产品或系统时的目标，如果实现了这些需求，将令客户满意
- 期望需求：隐含在产品或系统中，客户未能表达清楚的基本功能，缺少将会引起客户的不满
- 兴奋需求：超出客户预期的需求，另人非常满意

质量功能部署

- QFD通过客户访谈和观察、调查以及检查历史数据（如问题报告）为需求收集活动获取原始数据
- 把这些数据翻译成需求表——称为客户意见表，并由客户评审。
- 接下来使用各种图表、矩阵和评估方法抽取期望的需求并尽可能获取兴奋需求。

7.3.3 用户场景

- 收集需求时，系统功能和特性的整体愿景开始具体化
- 但是在软件团队弄清楚不同类型的最终用户如何使用这些功能和特性之前，很难转移到更技术化的软件工程活动中
- 为此，开发人员和用户可以创建一系列的场景
  - 场景可以识别对将要构建的系统的使用线索。
  - 场景通常称为用例，提供了系统将如何被使用的描述

7.3.4 获取工作产品

- 要求和可行性陈述。
- 系统或产品范围的界限说明。
- 参与需求导出的客户、用户和其他利益相关者的列表。
- 系统技术环境的说明。
- 需求列表以及每个需求适用的领域限制。
- 一系列使用场景，有助于深入了解系统或产品在不同运行环境下的使用。
- 任何能够更好地定义需求的原型。

- 7.3.5 敏捷需求获取
  - 用户故事
- 7.3.6 面向服务的方法

7.4 开发用例

- 用例说明对某个利益相关者的请求响应时，在各种条件下系统的行为
- 本质上，用例讲述了能表达主体场景的故事：
  - 最终用户如何在一特定环境下和系统交互。
  - 可以是叙述性的文本、任务或交互的概要、基于模板的说明或图形表示。
  - 用例从最终用户的角度描述了软件或系统。

开发用例

- 撰写用例的第一步是定义各类故事中所包含的“参与者”
- 参与者是在将要说明的功能和行为环境内使用系统或产品的各类人员（或设备）。
- 参与者是任何与系统或产品通信的事物，且对系统本身而言参与者是外部的。
- 每个参与者都有一个或多个目标

开发用例

- 参与者与最终用户并非一回事。
  - 用户可在使用系统时扮演了许多不同的角色，
  - 参与者表示了一类外部实体（经常是人员，但不限于人员），在用例中他们仅扮演一种角色。
- 例，考虑一个机床操作员（用户），他和生产车间（布置了许多机器人和数控机床）内的某个控制计算机交互。
- 在仔细考察需求后，控制计算机的软件需要四种不同的交互模式（角色）
  - 编程模式、测试模式、监测模式和纠错模式。
  - 4个参与者：程序员、测试员、监控员和故障检修员
  - 机床操作员可以扮演所有这些角色
  - 每个参与者的角色可能由不同的人员扮演。

开发用例

- 需求导出是一个逐步演化的活动，
  - 第一次迭代中并不能确认所有的参与者
  - 在第一次迭代中有可能识别主要的参与者
  - 对系统了解更多之后，才能识别出次要的参与者
- 主要参与者直接且经常使用软件，和他们交互可以获取所需的系统功能并导出系统的预期收益
- 次要参与者支持系统，以便主要参与者能够完成他们的工作。

开发用例

- 一旦确认了参与者，就可以开发用例。为了开发一个有效用例，可以考虑如下问题：
  - 谁是主要参与者、次要参与者？
  - 参与者的目标是什么？
  - 故事开始前有什么前提条件？
  - 参与者完成的主要工作或功能是什么？
  - 按照故事所描述的还可能需要考虑什么异常？
  - 参与者的交互中有什么可能的变化？
  - 参与者将获得、产生或改变哪些系统信息？
  - 参与者必须通知系统外部环境的改变吗？
  - 参与者希望从系统获取什么信息？
  - 参与者希望得知意料之外的变更吗？

SafeHome实例[4]

回顾基本的SafeHome需求，我们定义了4个参与者：**房主**（用户）、**配置管理人员**（类似房主，但扮演不同的角色）、**传感器**（附属于系统的设备）和**监测子系统**（监测SafeHome房间安全功能的中央站）。仅从该例子的目的来看，我们只考虑了房主这个参与者。房主通过使用报警控制面板或计算机等多种方式和住宅安全功能交互：

- 输入密码以便能进行其他交互。
- 查询安全区的状态。
- 查询传感器的状态。
- 在紧急情况时按下应急按钮。
- 激活/关闭安全系统。

SafeHome实例[5]

考虑房主使用控制面板的情况，系统激活的基本用例如下。⊖

1. 房主观察 SafeHome 控制面板（图 7-1），以确定系统是否已准备好接收输入。如果系统尚未就绪，“not ready”消息将显示在 LCD 显示器上，房主必须亲自动手关闭窗户或门以使得“not ready”消息消失。（not ready 消息意味着某个传感器是开着的，即某个门或窗户是开着的。）
2. 房主使用键盘键入 4 位密码，系统将该密码与已存储的有效密码相比较，如果密码不正确，控制面板将鸣叫一声并自动复位以等待再次输入，如果密码正确，控制面板将等待进一步的操作。
3. 房主选择键入“stay”或“away”（图 7-1）以启动系统。“stay”只激活外部传感器（内部的运动监控传感器是关闭的），“away”激活所有的传感器。
4. 激活时，房主可以看到一个红色的警报灯。

SafeHome实例[6]

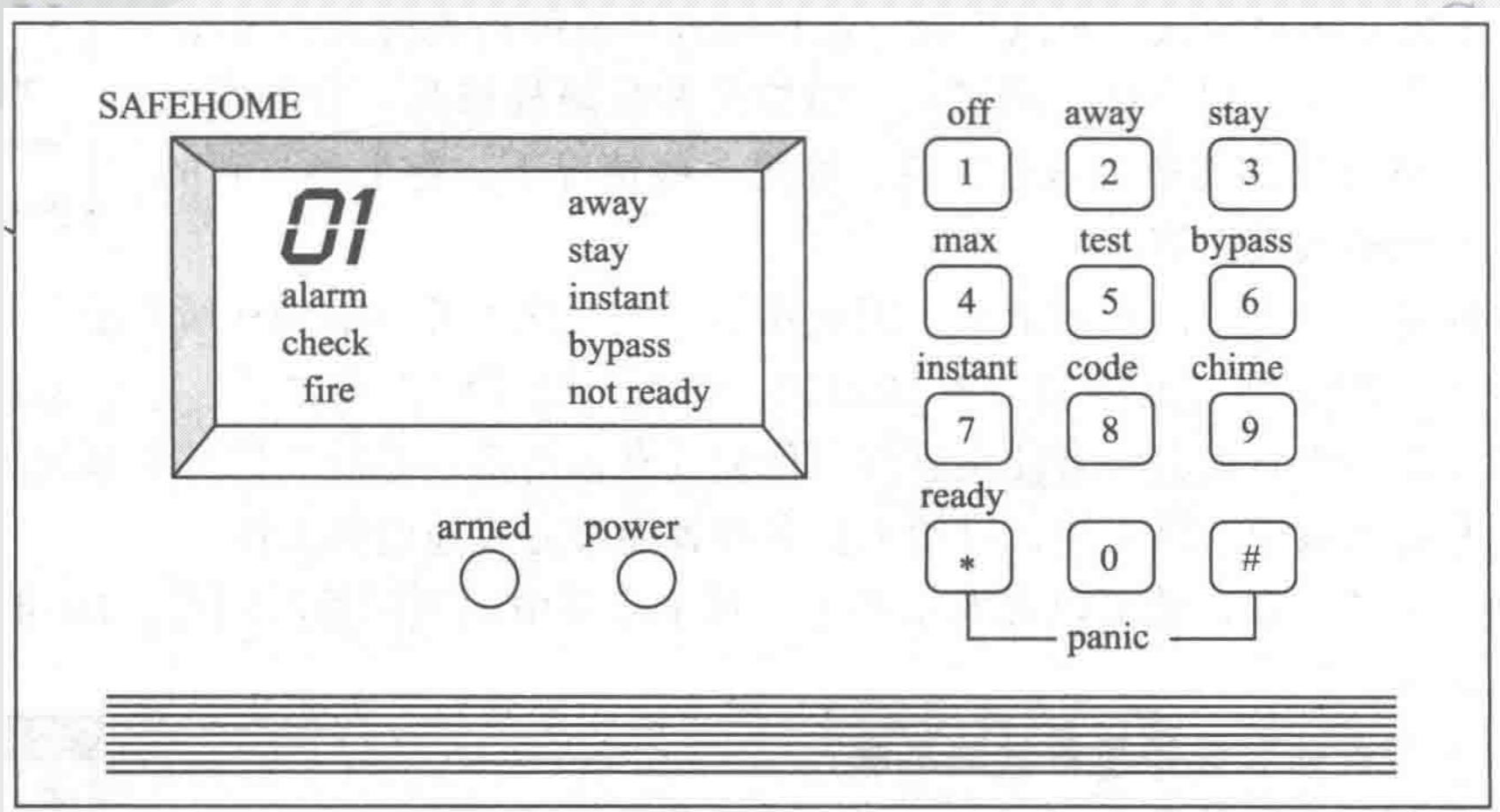


图 7-1　SafeHome 控制面板

SafeHome实例[7]

用例：初始化监控。

主要参与者：房主。

目标：在房主离开住宅或留在房间时，设置系统以监控传感器。

前提条件：系统支持密码输入和传感器识别功能。

触发器：房主决定“设置”系统，即打开警报功能。

场景：

1. 房主：观察控制面板。
2. 房主：输入密码。
3. 房主：选择“stay”或“away”。
4. 房主：观察红色报警灯显示 SafeHome 已经被打开。

SafeHome实例[8]

异常：

1. 控制面板没有准备就绪：房主检查所有的传感器，确定哪些是开着的（即门窗是开着的），并将其关闭。
2. 密码不正确（控制面板鸣叫一声）：房主重新输入正确的密码。
3. 密码不识别：必须对监控和响应子系统重新设置密码。
4. 选择 stay：控制面板鸣叫两声并且 stay 灯点亮；激活边界传感器。
5. 选择 away：控制面板鸣叫三声并且 away 灯点亮；激活所有传感器。

优先级：必须实现。

何时可用：第一个增量。

使用频率：每天多次。

使用方式：通过控制面板接口。

次要参与者：技术支持人员，传感器。

次要参与者使用方式：

技术支持人员：电话线。

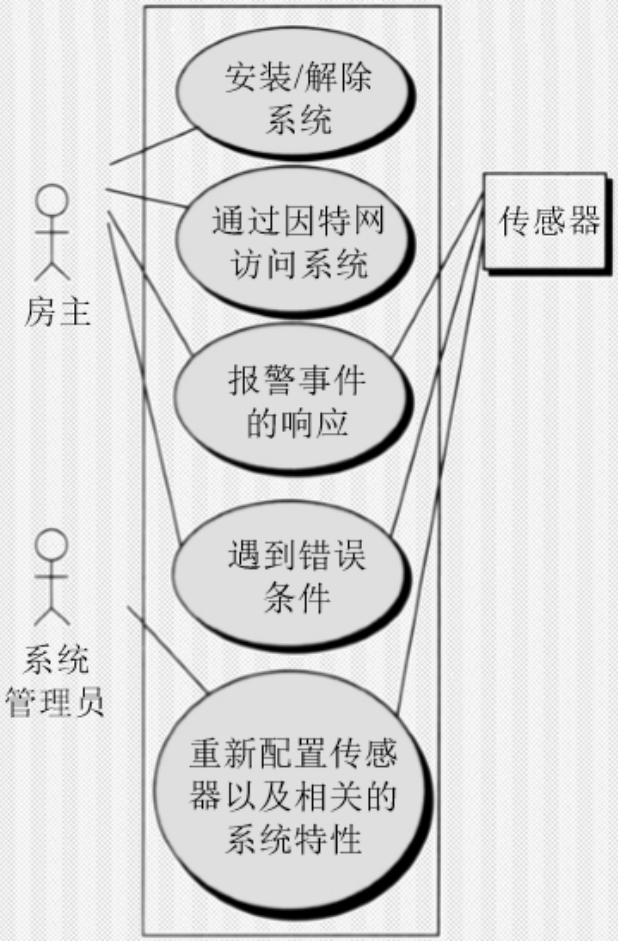
传感器：有线或无线接口。

SafeHome实例[9]

未解决的问题：

1. 是否还应该有不使用密码或使用缩略密码激活系统的方式？
2. 控制面板是否还应显示附加的文字信息？
3. 房主输入密码时，从按下第一个按键开始必须在多长时间内输入密码？
4. 在系统真正激活之前有没有办法关闭系统？

SafeHome实例[10]



7.5 构建分析模型

- 分析模型的目的是为基于计算机的系统提供必要的信息、功能和行为域的说明
- 随着软件工程师更多地了解将要实现的系统以及其他利益相关者更多地了解他们到底需要什么，模型将动态地变更。
- 当需求模型演化时，某些元素将变得相对稳定，为后续设计任务提供稳固的基础。
- 但是，有些模型元素可能是不稳定的，这表明利益相关者仍然没有完全理解系统的需求。

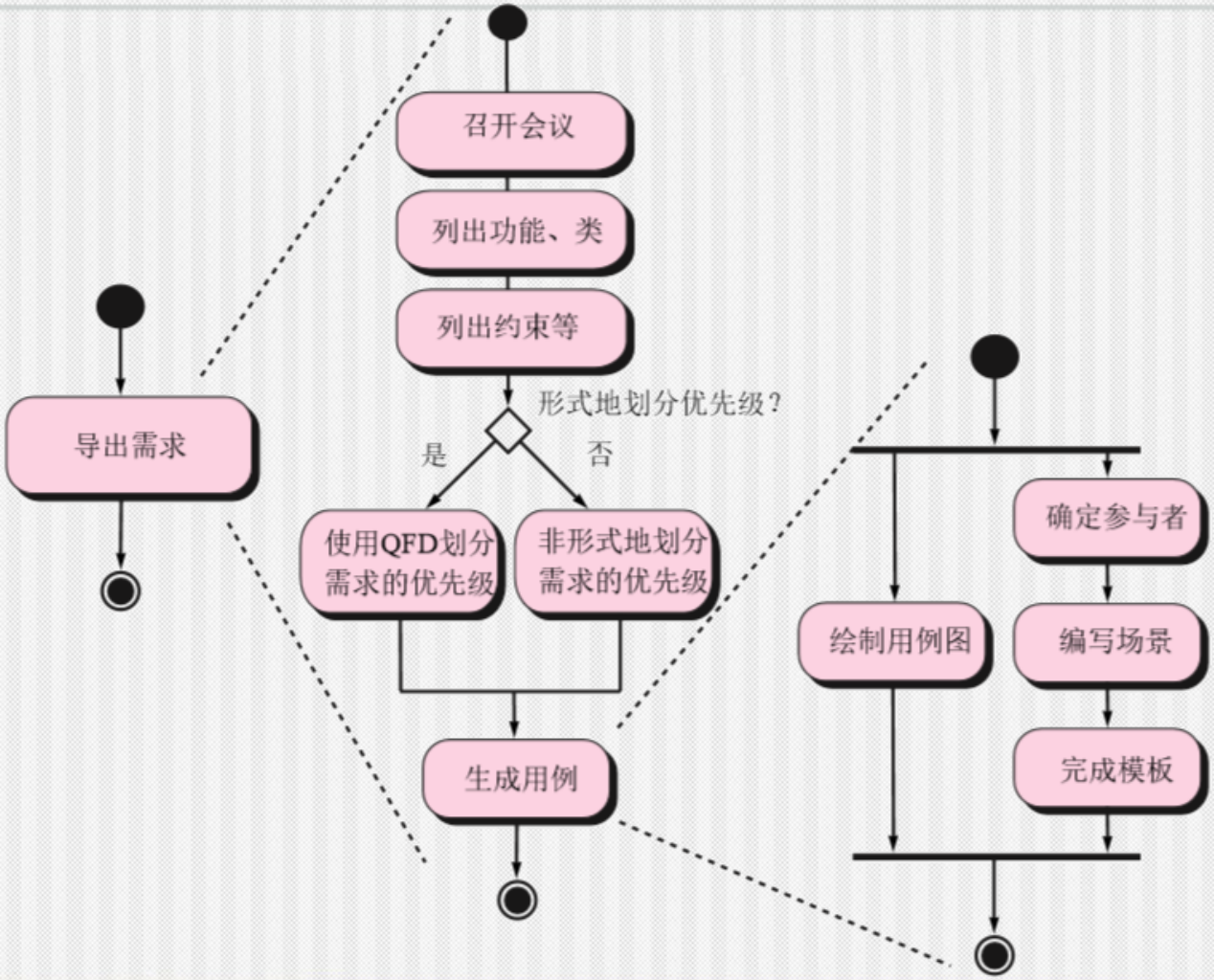
7.5.1分析模型的元素

- 有很多不同的方法考察计算机系统的需求。
- 某些软件人员坚持最好选择一个表达模式（如用例）并排斥所有其他模式。
- 有些专业人士则相信使用许多不同的表现模式来描述需求模型是值得的，不同的表达模式促使软件团队从不同的角度考虑需求——一种方法更有可能造成需求遗漏、不一致性和歧义性。

基于场景的元素

- 使用基于场景的方法可以从用户的视角描述系统
- 基本的用例及其相应的用例图演化成更精细的基于模板的用例。
- 课本图7-3描绘了一张使用用例引发的需求并表达它们的UML活动图。

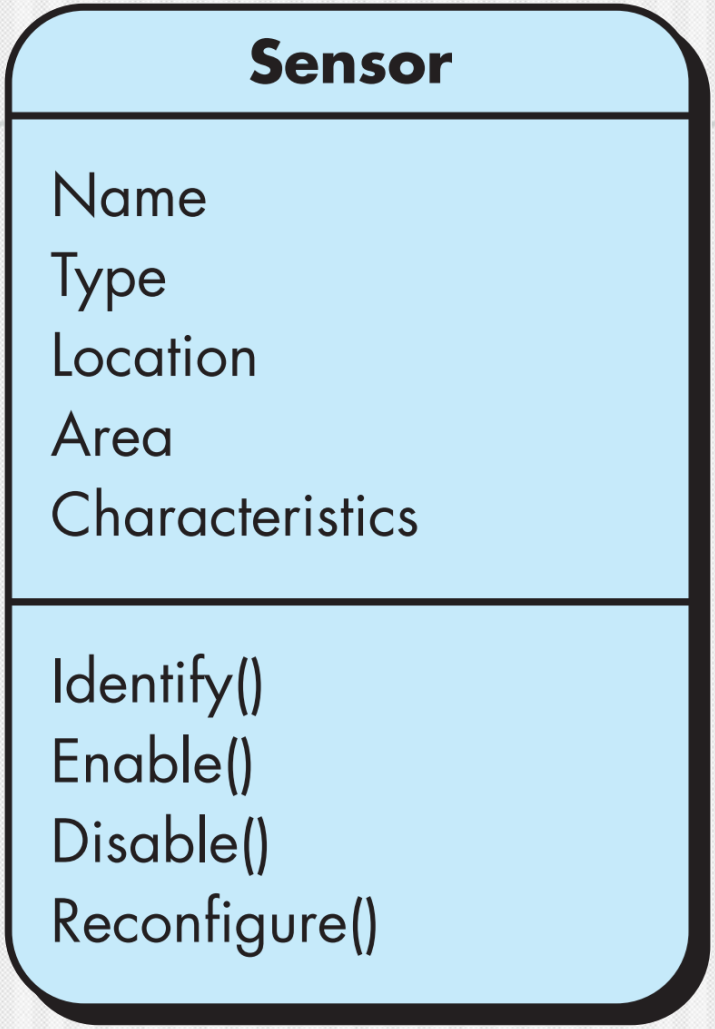
获取需求的UML活动图



基于类的元素

- 每个使用场景都暗示着当一个参与者和系统交互时所操作的一组对象，这些对象被分成类——
- 具有相似属性和共同行为的事物集合。
- 例如：可以用UML类图描绘SafeHome安全功能的传感器Sensor类如课本图7-4所示。
- 除了类图，其他分析建模元素描绘了类相互之间的协作以及类之间的关联和交互。

传感器Sensor的类图

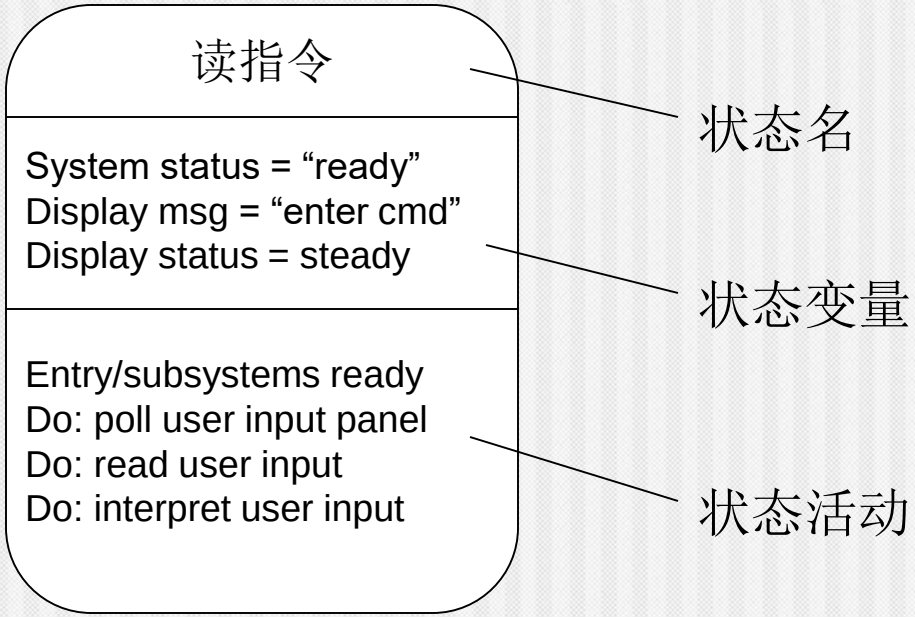


课本图7-4 传感器Sensor的类图

行为元素

- 基于计算机系统的行为能够对所选择的设计和所采用的实现方法产生深远的影响。因此，需求分析模型必须提供描述行为的建模元素。
- 状态图是一种表现系统行为的方法，
  - 该方法描绘系统状态以及导致系统改变状态的事件。
  - 状态是任何可以观察到的行为模式。
  - 状态图还指明了在某个特殊事件后采取什么动作。

UML状态图表示



课本图7-5 UML状态图表示

7.5.2 分析模式

- 7.5.2 分析模式
- 7.5.3 敏捷需求工程
- 7.5.4 自适应系统的需求

7.6 避免常见的错误

- 特性偏好
- 灵活性偏好
- 性能偏好

谢谢！