第4章. 需求获取概述

主要内容

1. 需求获取的非平凡性
2. 需求获取的活动过程
3. 需求获取活动的要点
4. 需求获取的技术

1. 需求获取的非平凡性

- 用户和开发人员的背景不同，立场不同
  - 首先是知识理解的困难。
    - 尽力去研究应用的背景，理解组织的状况，形成一个能够和用户进行有效沟通的粗略的知识框架
  - 默认（Tacit）知识现象
    - 利用有效的获取方法与技巧（角色扮演、观察等）来发现并获取默认知识

1. 需求获取的非平凡性

- 普通用户缺乏概括性、综合性的表述能力
  - 普通用户的知识结构就相对局限于一些具体的业务细节
    - 善于表达具体业务的细节问题
  - 专家用户的知识结构因其渊博性而具有概括性和广泛性
    - 能够回答概括性和综合性的问题
  - 开发人员在与用户接触之前就先行确定获取的内容主题，然后设计具体的应用环境和场景条件，由用户根据细节业务的执行来描述问题、表达期望。

1. 需求获取的非平凡性

- 用户存在认知困境
  - 潜在（Latency）知识
    - 需要利用各种有效的需求获取方法和技巧
- 用户越俎代庖
  - 用户提出的不是需求，而是解决方案
    - 注意保持业务领域和解决方案的区分界限
  - 用户固执的坚持某些特征和功能
    - 分析用户的深层目的，找到隐藏在背后的需求

1. 需求获取的非平凡性

- 缺乏用户参与
  - 用户数量太多，选择困难
  - 用户认识不足，不愿参与
  - 用户情绪抵制，消极参与
  - 没有明确的用户

  - 对系统的用户以及用户的替代源等相关涉众进行分析

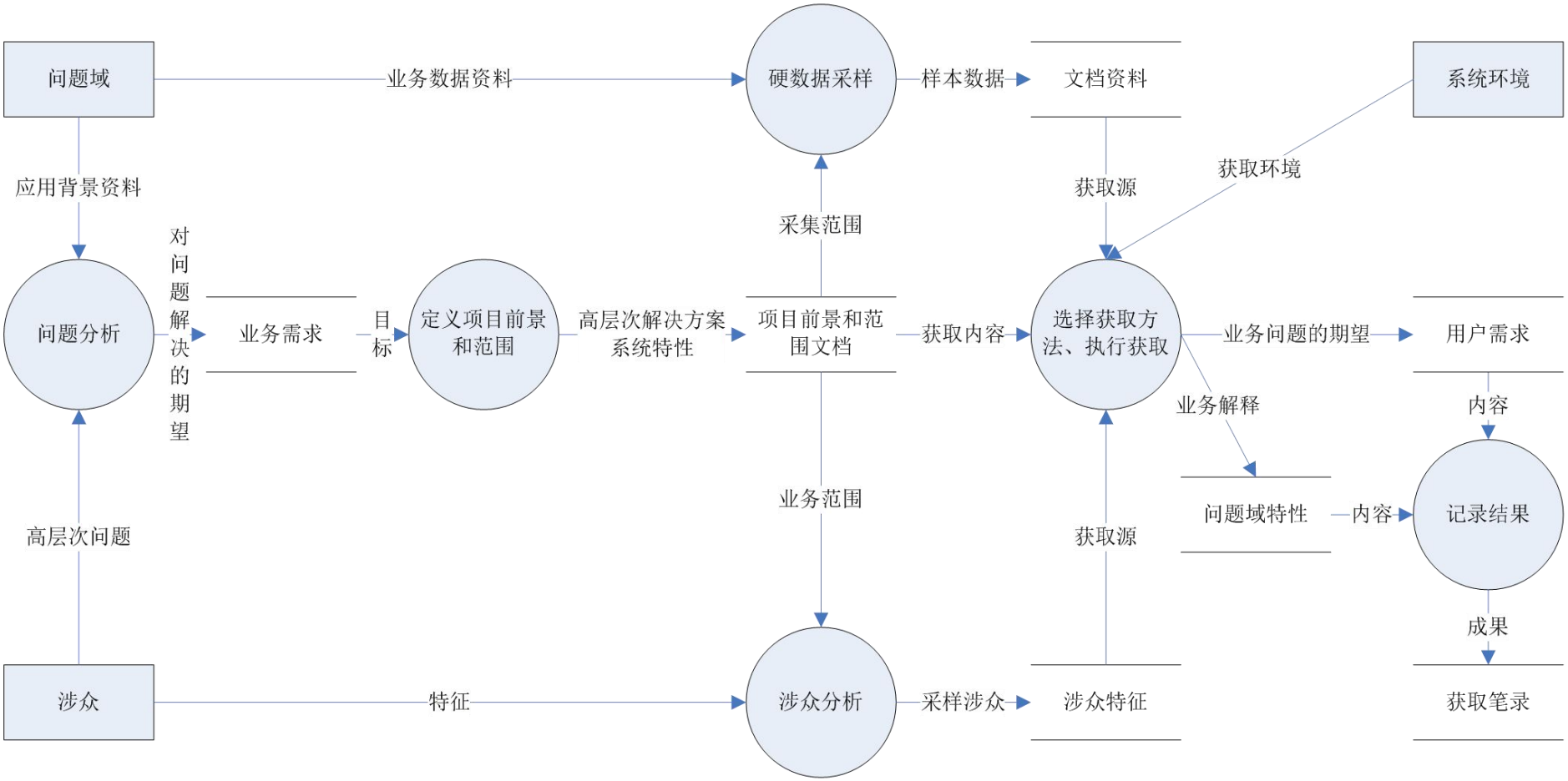
主要内容

1. 需求获取的非平凡性
2. 需求获取的活动过程
    1. 子活动
    2. 过程描述
3. 需求获取活动的要点
4. 需求获取的技术

2.1 需求获取的子活动

- 研究应用背景，建立初始的知识框架；
- 根据获取的需要，采用必要的获取方法和技巧；
- 先行确定获取的内容和主题，设定场景；
- 分析用户的高（深）层目标，理解用户的意图；
- 进行涉众分析，针对涉众的特点开展工作。

2.2 需求获取的活动过程



主要内容

1. 需求获取的非平凡性
2. 需求获取的活动过程
3. 需求获取活动的要点
    1. 获取的内容
    2. 获取的来源
    3. 获取的方法
    4. 获取的过程
    5. 获取的结果
4. 需求获取的技术

3.1 获取的内容

- 在项目的范围之内
- 所有为用户创建解决系统必须的信息
  - 需求
    - 通常体现为用户的观点、看法、目标或者问题
  - 问题域特性
    - 需要注意的是不要忽略系统的环境和约束
- 获取的内容不是一次得到的，而是逐步积累的

3.2 获取的来源

- 涉众
  - 用户
  - 客户
  - 领域专家
  - 市场人员、销售人员等其他用户替代源
- 相关产品
  - 原有系统
  - 竞争产品
  - 协作产品（和解系统存在接口的其他软件系统）
- 硬数据
  - 登记表格、单据、报表等定量文档
  - 备忘录、日志等定性文档
- 重要文档
  - 原有系统的规格说明
  - 竞争产品的规格说明
  - 协作产品的规格说明
  - 客户的需求文档（委托开发的规格说明、招标书）
- 相关技术标准和法规
  - 相关法律、法规及规章制度
  - 行业规范、行业标准

3.3 获取的方法

- 传统方法
  - 问卷调查、面谈、硬数据分析、文档检查、需求剥离等
- 集体获取方法
  - 头脑风暴（Brainstorming）、专题讨论会（Workshop）、JAD等
- 原型
- 认知方法
  - 任务分析（Task Analysis）、协议分析（Protocol Analysis）等
- 基于上下文的方法
  - 观察、民族志（Ethnography）和话语分析（Conversation Analysis）

3.4 获取的过程
——注意事项

- 在整体上制定组织方案
  - 确定系统的边界，建立上下文图或系统用例图
- 维护项目的前景和范围
- 引导和控制获取过程
- 接受需求的不稳定性
- 控制探索性工作

3.4 获取的过程
—— 防止需求遗漏

- 务必让所有的涉众都表达出自己的意见。
- 不要以抽象和模糊的需求作为结束。对抽象和模糊的需求，要进行细化，让真正的需求显露出来。
- 使用多种方法表达需求信息。利用不同的分析技术为相同的需求进行建模，通过分析不同的关注点，考察需求是否完整。
- 注意检查边界值和布尔逻辑。

3.4 获取的过程
——结束获取活动的判断条件

- 用户想不出更多的用例；
- 用户想出的新用例都是导出用例（通过其他用例的结合可以推导出该用例）；
- 用户只是在重复已经讨论过的问题；
- 新提出的特性、需求等都在项目范围之外；
- 新提出的需求优先级都很低；
- 用户提出的新功能都属于后继版本，而非当前版本

3.5 获取的结果

- 肯定会产生获取笔录（Elicitation Notes）
  - 用户需求、问题域知识和约束
  - 可能具有组织差、冗余、遗漏、自相矛盾等诸多问题
  - 可以包括文字记录、录音、摄像等各种形式
- 可能会产生两份定义明确的正式文档
  - 项目前景和范围文档
  - 用例文档

主要内容

1. 需求获取的非平凡性
2. 需求获取的活动过程
3. 需求获取活动的要点
4. 需求获取的技术

主要内容

1. 需求获取的非平凡性
2. 需求获取的活动过程
3. 需求获取活动的要点
4. 需求获取的技术
    1. 面谈法
    2. 观察和文档审查的方法
    3. 原型法
    4. 基于用例的方法

4.1 面谈法
—— 面谈中的问题

- 面对面的会见（face-to-face meeting）被认为是最具丰富内容的交流方法
- 实践当中应用最为广泛的需求获取方法之一
- 可以获得的信息内容包括
  - 事实和问题
  - 被会见者的观点
  - 被会见者的感受
  - 组织和个人的目标

4.1 面谈法
——面谈的类型

- 结构化面谈
  - 安全按照事先的问题和结构来控制面谈
- 半结构化面谈
  - 事先需要根据面谈内容准备面谈的问题和面谈结构，但在面谈过程当中，会见者可以根据实际情况采取一些灵活的策略
- 非结构化面谈
  - 没有事先预定的议程安排
  - 甚至会在没有太多事前准备的情况下就直接到访被会见者的工作地，就某个主题开展会谈
  - 会见者和被会见者谈话的主题可能非常广泛，而且每个主题都不会非常深入
  - 也可能在非结构面谈中仅就某个特殊的主题进行深入的讨论

4.1 面谈法
——面谈的优点和局限性

- 面谈的优点有：
  - 面谈的开展条件较为简单，经济成本较低；
  - 能获得包括事实、问题、被会见者观点、被会见者态度和被会见者信仰等各种信息类型在内的广泛内容；
  - 通过面谈，需求工程师可以和涉众（尤其是用户）建立相互之间的友好关系；
  - 通过参与面谈，被会见者会产生一种主动为项目做出贡献的感觉，提高涉众的项目参与热情。

4.1 面谈法
——面谈的优点和局限性

- 面谈的缺点和局限性包括：
  - 面谈比较耗时，时间成本较高；
  - 在被会见者地理分散的情况下往往难以实现面谈；
  - 面谈参与者的记忆和交流能力对结果影响较大，尤其是面谈的成功较高的依赖于需求工程师的人际交流能力；
  - 交谈当中常见的概念结构不同、模糊化表述、默认知识、潜在知识和态度偏见等各种问题在面谈中都不可避免，进而影响面谈的效果，导致产生不充分的、不相关的或者错误的数据；
  - 在会见者不了解被会见者认知结构的情况下，面谈不可能取得令人满意的效果。

4.1 面谈法
——相关方法

- 群体面谈
- 调查问卷
- 头脑风暴

4.1.1 群体面谈

- 群体面谈的方法是将所有的涉众方集中起来，选择一个合适的地点，集中一段时间，召开一个多方共同参与的会议，一起进行需求的讨论、分析和获取。
- 群体面谈的需求获取方法
  - 联合应用程序设计JAD（Joint Application Design）
  - 需求专题讨论会（Requirement Workshop）
  - 需求中心小组（Requirement Focus Group）
  - 联合需求规划JRP（Joint Requirements Planning）

5.1.1 群体面谈

- 优点：
  - 节约时间，有着更低的时间成本；
  - 在一个集中连续的时间内完成，能够加速项目的开发进度；
  - 涉众方可以直接交流，提高了冲突的处理能力和处理效率；
  - 这可以提高涉众的项目参与度；
  - 常常会有创造性的信息内容产生。
- 缺点：
  - 群体面谈要求所有参与方都要在一个集中的时间内抽出大量时间和精力投入面谈，这往往难以实现；
  - 群体面谈获得的信息比一对一面谈要复杂的多，因此对它们的分析是一个不小的技术挑战；
  - 主持群体面谈比主持一对一面谈要困难的多。

4.1.2 调查问卷

- 面谈方法以口头语言为主要的交流媒介，而调查问卷以文档为主要的交流媒介
- 适用调查问卷的情况
  - 系统的涉众在地理上是分布的；
  - 系统的涉众数量众多，而且了解所有涉众的统计倾向是非常重要的；
  - 需要进行一项探索性的研究，并希望在确定具体方向之前了解当前的总体状况；
  - 为后续的面谈标识问题和主题，建立一个开展工作的基础框架。

4.1.3 头脑风暴

- 它的目的不是发现需求，而是“发明”需求，或者说是发现“潜在”需求
- 它鼓励参与者在无约束的环境下进行某些问题的自由思考和自由讨论，以产生新的想法
- 适用情况
  - 发明并描述以前不存在的全新的业务功能
  - 明确模糊的业务
  - 在信息不充分的情况做出决策
- 包括两个阶段
  - **想法产生阶段**
  - **想法精减阶段**

4.1.3 头脑风暴
——想法产生阶段

- 目的是产生出尽可能多的新的想法
- 基本规则
  - 充分发挥想像力，不要有任何的羁绊；
  - 产生尽可能多的想法，想法重在数量而不是质量，不要顾及想法是否荒诞；
  - 自由讨论，目的是产生新的想法，不要争吵和批评；
  - 在自由讨论当中，可以转换和组合所有已提出的想法，以产生新的想法
- 通常持续1个小时左右，特殊情况下持续2～3个小时

4.1.3 头脑风暴
——想法精减阶段

- 第一步是去除那些不值得进一步讨论的想法
- 第二步是把类似的意见进行归类
- 第三步是主持人遍历每一个未被删除的想法，确保所有参与者都对其有共同的理解
- 利用投票或类似方法，评估现有想法的优先级
- 最后，根据评估的数据，从中筛选出符合一定标准的想法作为头脑风暴方法的成果

4.2 观察和文档审查的方法

- 应用于用户无法完成主动的信息告知的情况下
- 某些事件只有和它们发生时的具体环境联系起来，才能得到理解
  - 企业流程
  - 企业文档
  - 业务表格/报告
  - 已有系统的运行情况

4.2 观察和文档审查的方法

- 优点:
  - 理解复杂的协同事件
  - 获取工作中的异常处理
  - 获取与用户认知不一致的实际知识
  - 了解用户的认知
  - 获取默认（tacit）知识
- 缺点:
  - 获得的是零散的细节知识,需要归纳整理
  - “假象”

4.3 原型法
——什么是原型

- “原型是一个系统，它内化了（capture）一个更迟系统（later system）的本质特征。原型系统通常被构造为不完整的系统，以在将来进行改进、补充或者替代。”
- 如果在最终的物件（final artifact）产生之前，一个中间物件（mediate artifact）被用来在一定广度和深度范围内表现这个最终物件，那么这个中间物件就被认为是最终物件在该广度和深度上的原型。

4.3 原型法
——原型的类别

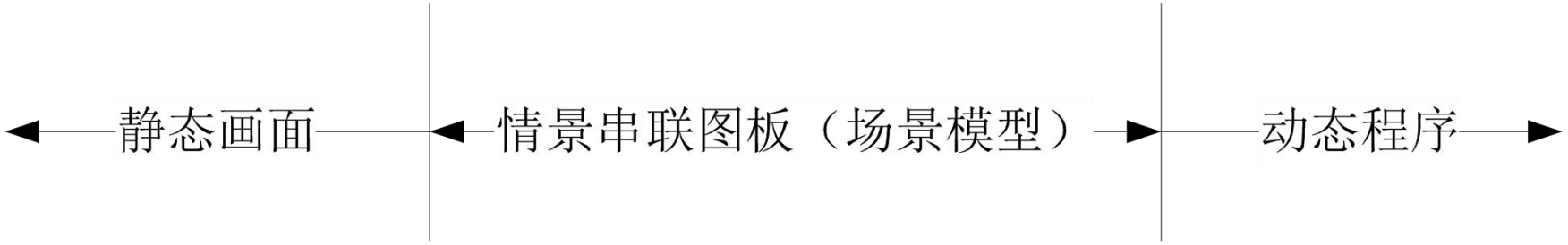
- 按照使用方式分类
  - **演示原型（presentation prototype）**
    - 主要被用在启动项目阶段
    - 目的是让用户相信应用系统的开发是可行的
  - **严格意义上的原型（prototype proper）**
    - 主要被用在分析需求阶段
    - 用来阐明用户界面或者系统功能的某些特定方面
  - **试验原型（breadboard prototype）**
    - 主要被用在构建系统阶段
    - 帮助开发者澄清他们所面对的一些和系统构建相关的技术问题
  - **引示系统原型（pilot system prototype）**
    - 会被开发在系统开发的各个阶段
    - 用作最终系统的构建核心

4.3 原型法
——原型的类别

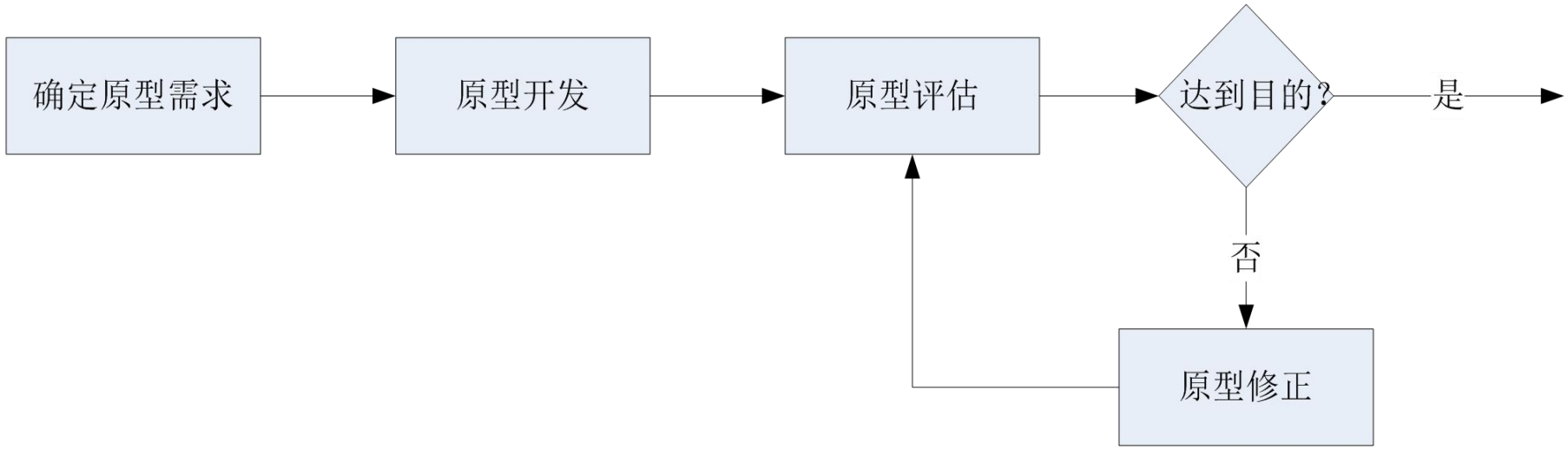
- 按照开发方法分类
  - **探索式（exploratory）**
    - 以缺陷需求开始继而不断调整和修正需求的原型开发方式称为探索式
    - 要尽可能的调整各种设计选项
  - **实验式（experimental）**
    - 以清晰的用户需求和模糊的实现方法、实现效果、可行性开始，明确需求的可行性和技术实现方案
    - 定义一个对原型的评估方法，确定评估的属性
  - **演化式（evolutionary）**
    - 以清晰的原型化需求和项目积累下来的原型资产为开始
    - 原型化的需求，也有项目积累下来的原型资产
    -

4.3 原型法
——原型的类别

- 按照表现分类



4.3 原型法
——原型方法的过程



4.3 原型法
——原型方法的风险

- 涉众看到了一个正在运行的原型，得出产品几乎已经完成的结论，从而提出快速交付产品的不当要求
- 用户可能会被原型所表现出来的非功能特性遮蔽了眼睛，从而忽略了他们更应该重视的功能特性
- 在澄清需求不确定性的同时也可能会掩盖一些用户的假设，这些假设将会无从发现
- 原型开发工作投入太多的工作，使得开发团队消耗了过多的时间和过大的成本

4.4 基于用例的方法

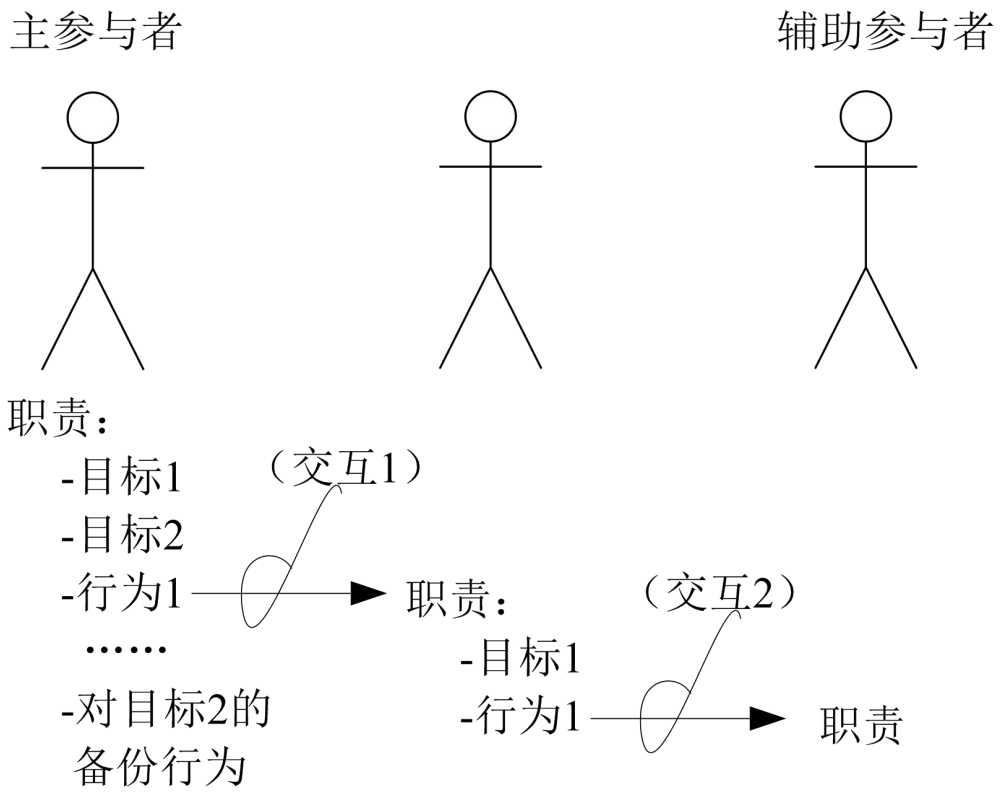
- 用例
  - 相关场景集合的叙述性的文本描述
  - 用例的概念是[Jacobson1992]最先在Objectory方法中提出的
  - UML以用例来捕获系统的功能需求，而不是所有的系统需求
  - 被广泛应用：以用例为中心

4.4 基于用例的方法

- UML将用例定义为“在系统（或者子系统或者类）和外部对象的交互当中所执行的行为序列的描述，包括各种不同的序列和错误的序列，它们能够联合提供一种有价值的服务”[Rumbaugh2004]。
- [Cockburn2001]认为用例描述了在不同条件下系统对某一用户的请求所作出的响应。根据用户的请求和请求时的系统条件，系统将执行不同的行为序列，每一个行为序列被称为一个场景。
- 目标性，多场景性
- 功能需求（擅长）与非功能需求（偏弱）

4.4 基于用例的方法

- 重要术语



4.4 基于用例的方法
——用例的描述

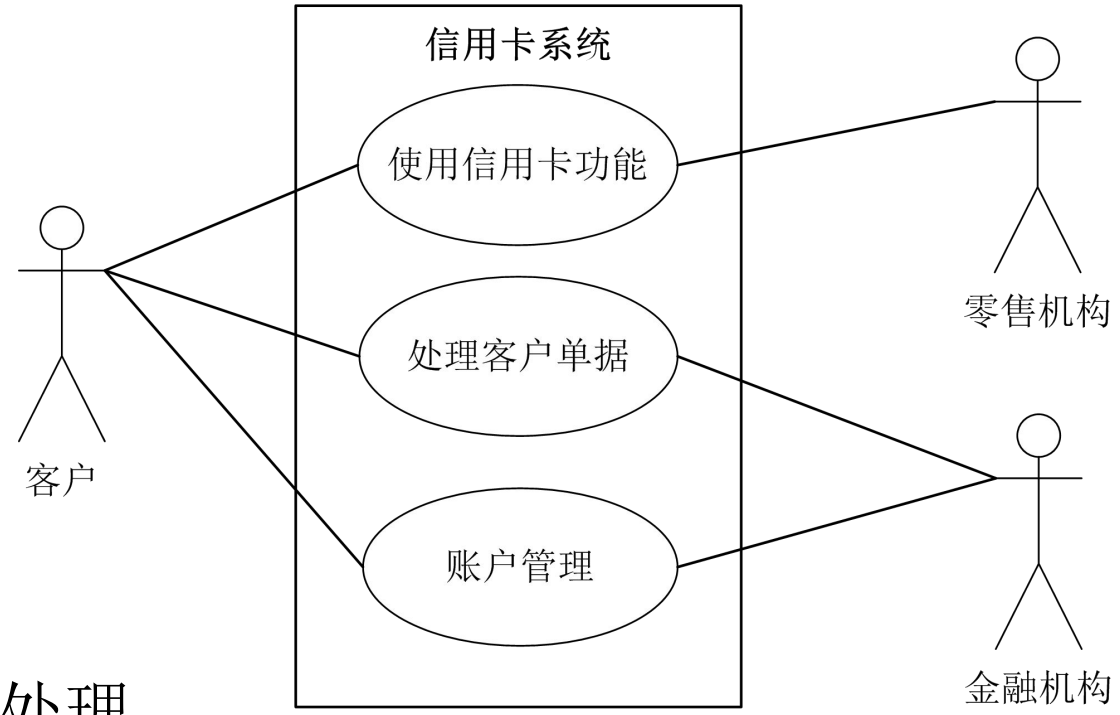
ID:	用例的标识，通常会结合用例的层次结构使用X.Y.Z的方式
名称:	对用例内容的精确描述，体现了用例所描述的任务，通常是“动词＋名词”
用例属性	包括创建者、创建日期、更新历史等
参与者:	描述系统的主参与者、辅助参与者和每个参与者的目标
描述:	简要描述用例产生的原因，大概过程和输出结果
优先级:	用例所描述的需求的优先级
触发条件:	标识启动用例的事件，可能是系统外部的事件，也可能是系统内部的事件，还可能是正常流程的第一个步骤
前置条件:	用例能够正常启动和工作的系统状态条件
后置条件:	用例执行完成后的系统状态条件
正常流程:	在常见和符合预期的条件下，系统与外界的行为交互序列
分支流程:	用例中可能发生的非常见的其他合理场景
异常流程:	在非预期的错误条件发生时，系统对外界进行响应的交互行为序列
相关用例:	记录和该用例存在关系的其他用例。关于用例之间的关系见10.4.4
业务规则:	可能会影响用例执行的业务规则
特殊需求:	和用例相关的其他特殊需求，尤其是非功能性需求
假设:	在建立用例时所做的假设
待确定问题:	一些当前的用例描述还没有解决的问题

4.4 基于用例的方法

- 用例的发现
  - 涉众及其目标
  - 每个涉众的一个目标意味着一个用例
- 用例的使用
  - 描述、探索与解释

4.4 基于用例的方法

- 用例模型
  - 用例
  - 参与者
  - 关联
  - 系统边界
  - 多用例综合处理
    - 不允许功能分解



本章小结

- 需求获取是一个困难和复杂的任务
- 需求获取的成功执行需要有效组织子活动过程
- 执行需求获取时既要尽可能全面，又要防止不完备，更要注意进行过程控制
- 常见的需求获取技术及其适用情境