# 3.2 需求获取与建模

- 需求获取的挑战与途径
  - 需求获取的目标
    - 收集准备建立的系统和正在使用的系统的信息，并从这些信息中提取用户和系统需求
    - 为下一步的需求分析提供素材
  - 需求获取的基本步骤
    - **了解相关背景和领域/行业的知识**，确定产品所期望的**用户类**
    - 与客户企业或组织的高层人员进行**交流**，了解实际用户**任务和目标**以及这些人物所支持的**业务需求**
    - 与客户企业或组织的底层人员进行**交流**，获取**每个用户类的详细的用户需求**
    - **整理需求既要，发现新问题**，重复前三步
    - **需求分类和组织**，以区分**功能需求、非功能需求、约束条件、业务规则、外部接口需求**、建议解决方案和附加信息
    - **优先排序和冲突解决**
    - 得到最终需求清单，并与客户做**最终签字确认**
  - 需求获取面临的困难

| 问题 | 解决方案 |
| --- | --- |
| “Yes, But”综合症：直到开发人员将用户描述的东西交给他们，用户才认为他们知道自己要什么 | 尽早提供可选择的启发技术原型开发、迭代方法等 |
| “Undiscovered Ruins”综合症 用户不知道自己需要什么，或知道但不知如何表达 | 详细调研，将用户当作领域专家来认识和激励，尝试其他交流和启发技术 |
| “User and Developer”综合症 双方在沟通时存在交流的鸿沟 | 熟悉应用领域，把分析员放在用户的位置上，采用用例分析方法 |

  - 需求获取的手段
    - 面对面访谈
      - 优点
        - 获取大量丰富的数据，有助于发现观点、感受、目标及事实
        - 可以进行深入探讨，根据面谈前一阶段获知的内容调整后续的问题
      - 缺点
        - 数据是定性的，难以分析
        - 难于对问题的回答者进行比较
        - 面谈技巧难于掌握
    - 问卷调查

        - 使用场景
            - 涉众广泛
            - 有预先良好定义的问题，且需要答案
            - 希望得到科学的统计分析结果
            - 验证有限次面谈得出的结论
        - 优点
            - 可以快速获取的大量反馈
            - 可以远程执行
            - 可以搜集关于态度、信念及特性的信息
        - 缺点
            - 简单的分类导致对上下文的考虑较少
            - 留给用户自由表达的空间较小
            - 样本存在偏见
            - 样本规模太小
            - 自由发挥问题难以分析
    - 专题讨论会
        - 目标
            - 介绍项目成员和干系人
            - 收集需求列表
            - 在会议过程中，运用头脑风暴、故事板、角色扮演、现有系统评估等方法获需求
        - 指导原则
            - 给每个人发言的机会
            - 保持研讨话题的相关性
            - 定义需求属性
            - 记录需求发现
            - 总结并作出结论
    - 头脑风暴
        - 目标
            - 通过群组效应，激发新产品和系统的新想法
            - 在需求不完全明确的情况下比较有用
        - 指导方针
            - 采用有组织的研讨会形式
            - 百花齐放，不评价、不争论、不批评
            - 不受现实可行性限制
            - 新观点多多益善

- 抛砖引玉
- 互相启发

- 需求建模
  - 基于场景的方法
    - 用户故事
      - 用户故事从用户角度对功能的简要描述（也可以描述非功能需求）
      - 用户故事卡片格式：作为一个 `角色` ，可以 `活动` ，以便于 `价值`
      - 好的用户故事应具备的特征
        - 独立性（Independent）：尽可能避免故事之间存在依赖关系，否则会产生优先级和规划问题。
        - 可协商（Negotiable）：故事是可协商的，不是必须实现的书面合同或者需求。
        - 有价值（Valuable）：确保每个故事对客户或者用户有价值的，最好是让用户编写故事。
        - 可估算（Estimatable）：开发者应该能够预测故事的规模，以及编码实现所需要的时间。
        - 短小的（Small）：故事尽量短小，最好不超过10个理想人天，至少在一个迭代中完成。
        - 可测试（Testable）：所编写的故事必须是可测试的。
      - 3C
        - 卡片（Card）：故事的简短描述
        - 交谈（Conversation）：和客户或者产品负责人的交流沟通，用户如何与系统交互
        - 确认（Confirmation）：如何通过验收测试确认用户故事被正确完成
      - 注意事项
        - 用户要进行分类
        - 给出来的 user story 需要有价值
        - User Story 目标过多，导致功能过多
        - User Story 的目标和功能不匹配
        - 描述 User Story 的文字太主观
    - 用例图
      - 统一建模语言 UML
        - 用来可视化、描述、构造和文档化软件密集型系统的各种产品
        - 支持不同人员之间的交流
      - 基于用例的需求获取
        - 什么是用例

          用例描述了使用者使用系统完成用户某一目标的过程
          用例把系统描述为实现特定客户目标所要做的事

一个用例聚焦于**单一的目标**
一个用例可能**包含多个功能**

- 什么是用例图

用例图包含系统的所有用例
**用例图是系统的蓝图**

- 基本概念
  - 参与者

    是某些具有行为的事物，可以是人、计算机系统或者组织，且对系统本身而言参与者是外部的
  - 场景

    是参与者和系统之间的一些列特定的活动交互，也称为用例实例
  - 用例

    就是一组相关的成功和失败的场景集合，用来描述参与者如何使用系统来实现其目标
- 特征
  - 行为序列

    一个用例由一组可产生某些特定结果的行为产生，这些行为是不可再分解的
  - 系统执行

    系统为外部角色提供服务
  - 可观测到的、有价值的结果

    用例必须对用户产生价值
  - 特定的角色

    某人、某台设备、某外部系统等能够出发某些行为
- 基本思想

  从用户的角度来看，他们并不想了解系统的内部结构和设计，他们所关心的是系统所能提供的服务，也就是被开发出来的系统将是如何被使用的
- 组成元素
  - 参与者
  - 用例

    用例从较高层次上给出参与者和系统之间交互的故事
  - 通讯关联

    用于表示参与者和用例之间的对应关系，它表示参与者使用了系统中的哪些服务(用例)、系统所提供的服务(用例)是被哪些参与者所使用的
- 优点
  - 系统被看成是一个黑箱，不关心系统内部是如何完成其所提供的功能
  - 描述了被定义系统由哪些**外部使用者**，其与被定义系统发生交互
  - 针对每一参与者，描述了系统为这些参与者提供了**什么样的服务**
- 绘制用例模型的步骤
  - 确定系统边界

        - 识别并描述参与者
        - 确定每个参与者目标，识别用例
        - 势必额参与者与用例之间的通讯关联
        - 给出每一个用例的详细描述
        - 细化用例模型
            - 参与者与参与者，用例与用例之间的泛化
            - 用例和用例之间的包含
            - 用例和用例之间的扩展
    - 最终的提交物
        - 用例模型
        - 每个用例的详细描述
        - 术语表：所用到的术语说明
        - 补充归约：非功能性需求的说明
- 活动图
    - 显示了组成复杂过程的步骤序列，例如算法或工作流
    - 用于详细描述样例
    - 是状态图的一个变种
    - 目的是描述动作及动作的结果
    - 活动图中的动作可以放在“泳道”，泳道聚合一组活动
- 基于类的方法
- 基于模式的方法