# 3a - Requirements

## 需求工程

### **第3讲**

讲师：Ruzanna Chitchyan、Jon Bird、Pete Bennett

## 概览

* 什么是需求？
* 利益相关方识别
* 敏捷开发中的需求
* 使用 UML 表达需求
  + 用例图
  + 用例说明

## 需求

* 系统需求**不以系统实现的形式**定义系统，而是**以用户观察的形式**定义系统。需求记录了系统的特性和限制的描述。
  + **功能需求**定义了用户与系统的交互，说明系统应该做什么：
    - 系统应提供的服务说明。
    - 系统对特定输入的反应。
    - 系统在特定情况下的行为。
    - 可能还包括系统不应做的事情。
  + **非功能需求**定义了系统的其他属性，说明如何实现功能需求：
    - 系统服务或功能的限制。
    - 通常适用于整个系统，而不仅仅是个别特性。

## 为什么需要需求工程？

需求工程解决了以下问题：

* **术语不一致**：人们会用自己的语言表达需求。
* 对同一系统的需求可能存在**冲突**。
* 人们**经常不知道自己想要什么**（或者至少无法清楚地解释）。
* 需求**经常发生变化**。
* 相关人员或信息可能**无法获取**。

## 需求的作用

### **需求是沟通的机制**

需求在不同角色之间传递信息：

* 客户
* 系统管理员
* 测试员
* 开发人员
* 业务分析师

### **需求是指令**

* 说明应该**做什么**！

### **需求是验收标准**

* 为了公平评估团队是否交付了符合需求的产品，需求必须：
  + 明确且精确。
  + 足够易于获取和可衡量。
  + 易于理解且清晰。

## 需求驱动敏捷流程

在敏捷开发中，需求与以下角色密切相关：

* 客户
* 测试员
* 产品负责人
* 开发人员

## 分析需求

### **主要步骤：**

1. 识别与系统相关的**利益相关方**。
2. 确定用户的**顶层需求**（如非功能需求或宏观目标/史诗级需求）。
3. 将需求**细化为具体的用户故事**。
4. 制定**原子需求**（如正式规范）。
5. 使用一些 UML 工具，例如用例图和用例描述。

## 识别利益相关方

### **洋葱模型**

* **外围环境**：例如，政府监管者、公众、政治家等。
* **包含系统**：例如，维护人员、正常操作员、功能受益者等。
* **系统内部**：例如，开发人员、操作支持人员。

#### **开发人员的位置：**

* 在开发阶段，开发人员与产品关系密切。
* 在产品投入服务后，开发人员可能仅参与维护角色。
* 根据产品开发阶段，可绘制不同的洋葱模型。

## 识别顶层需求/关注点

### **史诗级需求（Epics）**

* 高层次需求（没有细节）。
* 以业务或用户价值为重点。
* 范围大且复杂。
* 需要分解为更小的部分实施。
* 所有子需求均归属于该史诗级需求。

**示例：**

* 提高视觉可访问性。
* 实现替代输入方法。
* 构建可访问性设置选项。

**整体目标（Initiative）：**

* “提高残障用户的可用性”。

## 将需求分解为更小步骤/细化需求

### **用户故事模板**

用户故事的标准格式为：

* **作为一个 <用户类型>，我希望 <实现目标>，以便于 <达成原因>。**

**示例：** 史诗级需求：提高视觉可访问性

* 作为低视力用户，我希望调整文本大小，以便我可以舒适阅读。
* 作为色盲用户，我希望自定义配色方案，以便无论颜色感知如何都能看到内容。
* 作为用户，我希望文本与背景之间有足够的对比度，以便能清晰阅读应用程序的所有部分。

## 创建良好用户故事的标准（INVEST）

* **独立性（Independent）**：可以独立于其他用户故事进行开发。
* **可协商性（Negotiable）**：灵活且开放讨论。
* **有价值性（Valuable）**：为用户提供明确价值。
* **可估算性（Estimable）**：可估算工作量。
* **小型化（Small）**：足够小以在一个迭代周期内完成。
* **可测试性（Testable）**：具有明确的完成标准。

## 用户故事的验收标准

### **好的验收标准应具备：**

* **清晰性**：易于理解且不含歧义。
* **可测试性**：可以通过测试验证。
* **可衡量性**：可以定量或定性地测量。
* **原子性**：每个标准都是独立的，可单独检查。

### **模板：**

* **Given**（给定某种初始条件或前提），**When**（当执行某个动作或事件），**Then**（预期结果）。

**示例验收标准：** 用户故事：作为低视力用户，我希望调整文本大小以便于舒适阅读。

1. 文本大小调整：给定我在网站或应用程序中，当我导航到可访问性设置时，我应看到一个调整文本大小的选项（例如小、中、大、特大）。
2. 对非文本元素无影响：给定我调整文本到任意大小，当我查看非文本元素（如图像或图标）时，它们不应变形或改变大小。
3. 默认重置选项：给定我已调整文本大小，当我想恢复默认大小时，我应看到并能使用“重置为默认”选项。

## 宏观目标 vs 史诗级需求 vs 用户故事

| **宏观目标（Initiative）** | **史诗级需求（Epic）** | **用户故事（User Story）** |
| --- | --- | --- |
| 公司战略目标，具有重要的业务成果。 | 大型的、战略性的目标。 | 特定的功能或特性。 |
| 涉及多个史诗级需求及团队/部门。 | 涉及多个迭代周期（sprints）。 | 在一个迭代周期内完成。 |
| 示例：“改善残障用户的体验。” | 示例：“提升视觉可访问性。” | 示例：“作为一名低视力用户，我希望调整文本大小以便舒适阅读。” |

## 用户故事优先级：实施什么？

### **考虑以下因素：**

* 用户需求。
* 业务价值。
* 技术可行性。

### **MoSCoW 优先级划分：**

* **必须有（Must-Have）**：至关重要的需求。
* **应该有（Should-Have）**：重要但非关键的需求。
* **可以有（Could-Have）**：额外加分项。
* **不会有（Won’t-Have）**：当前范围之外的需求。

### **价值与工作量评估：**

* 高价值，低工作量（优先实施）。
* 高价值，高工作量（次优先）。
* 低价值，低工作量（有时间可实施）。
* 低价值，高工作量（尽量避免）。

## 非功能需求

非功能需求定义了系统的整体属性，例如：

* 性能。
* 可扩展性。
* 可靠性。
* 可维护性。
* 安全性。

## UML 简介

### **什么是用例模型？**

* **系统行为**是指系统的动作与反应。
* 用例描述了系统与（部分）环境之间的交互。
* 描述系统的功能需求，连接利益相关方需求与软件需求。
* 用作规划工具。
* 包含以下元素：
  + \*\*演员（Actor）：\*\*代表与系统交互的角色。
  + \*\*用例（Use Case）：\*\*描述演员和系统之间的交互步骤。

## 用例规格说明

用例规格说明是一个描述特定用例的需求文档，包含以下内容：

* \*\*事件流程：\*\*描述演员与系统之间的交互。
* **其他信息：**
  + \*\*前置条件：\*\*用例执行前必须满足的条件。
  + \*\*后置条件：\*\*用例执行完成后系统的状态。
  + \*\*特殊需求：\*\*对系统性能或设计的具体要求。
  + \*\*关键场景：\*\*最常见或最重要的操作序列。
  + \*\*子流程：\*\*用例中嵌套的流程。

## 用例的流程结构

用例流程分为两种：

1. \*\*基础流程：\*\*从开始到结束的主要场景（成功执行的理想流程）。
2. \*\*替代流程：\*\*包括以下几种情况：
   * \*\*常规变体：\*\*正常流程的不同变体。
   * \*\*特殊情况：\*\*与常规流程不同的特殊处理。
   * \*\*错误流程：\*\*处理异常或错误。

## 用例检查点

确保以下条件：

1. 每个用例相互独立，不与其他用例有直接重叠。
2. 没有两个用例具有非常相似的行为或流程。
3. 用例中任何部分的流程都未在其他用例中重复建模。

## 总结复习

请确保理解以下问题：

* **模型的用途是什么？**
* **什么是系统行为？**
* **什么是演员（Actor）？**
* **什么是用例（Use Case）？**
* **如何判断需求的质量是否良好？**