Certified Professional for Requirements Engineering

需求工程@敏捷基础

教学大纲和学习指南

Lars Baumann, Peter Hruschka,
Kim Lauenroth, Markus Meuten,
Sacha Reis, Gareth Rogers,
François Salazar,
Hans-Jörg Steffe, Thorsten Weyer

v1.4.0 | 7 月, 1 日, 2024

使用规则

1. 培训机构及个人可以引用这份教学大纲和学习指南作为培训的依据，但必须注明本文的出处及版权。任何以广告目的引用本文的行为需要得到 IREB 的书面许可。
2. 任何团体或者个人在发表文章、出版书籍或者其它的发表性刊物中引用本文的，但必须注明出处和作者以及 IREB 的版权所有。

鸣谢

本教学大纲及学习指南作者为： Lars Baumann, Peter Hruschka, Kim Lauenroth, Markus Meuten, Sacha Reis, Gareth Rogers, François Salazar, Hans-Jörg Steffe, Thorsten Weyer。

本教学大纲和学习指南的审校为： Bernd Aschauer, Thomas Emmerich, Dirk Fritsch, Rainer Grau, Andrea Hermann, Krystian Kaczor, Niko Kaintantzis, Elisabeth Larson, Ladislau Szilagyi, Daniel Tobler, Erik van Veenendaal, Arun Vetrivel, Sven van der Zee。

参加本次本地化的 CSTQB 专家有（按姓氏拼音排序）：包晓芳、崔哲、范臻玉、高建民、刘海英（组长）、覃时悦（责任评审）、商超博、苏国霞、文晓慧、夏勇（责任评审）、张晓峰、周震漪（责任评审）。

感谢所有对本文做出贡献的人。

经 Xavier Franch 推荐，IREB 委员会于 2017 年 3 月 2 日批准了该大纲和学习指南的英文版本的发布。

本文的用途

本教学大纲和学习指南定义了“需求工程@敏捷”基础级所需要的知识水平，由 IREB（国际需求工程委员会）拟定。本教学大纲和学习指南为培训机构制定课程教材提供了基础。学生可以使用本教学大纲和学习指南为参加考试作准备。

教学大纲和学习指南的内容

基础级别是根据所有涉及需求工程和敏捷开发专业人员的需要而编写的。包括的人员角色，例如项目管理人员/IT 管理人员，领域专家，系统分析师，软件开发人员，敏捷专家，产品负责人，以及作为敏捷组织中的相关人员。

内容范围

需求工程@敏捷源于这两个观点：从 IREB 的角度看敏捷的价值，以及从敏捷的角度看需求工程的价值。其内容包括对敏捷环境中的需求工程制品和技术、需求工程环境中的敏捷制品和方法以及敏捷产品开发中的基本过程元素的分类和评估。需求工程@敏捷指出了在开发过程中使用敏捷的动机。

一个非常重要的主题是需求工程与敏捷直接的协同作用：与需求工程相关的敏捷原则以及与需求工程价值相关的敏捷思维。

IREB 需求工程@敏捷认证旨在帮助以下人群：

- 希望参与到敏捷开发以及希望成功地将他们的技术应用到敏捷环境的需求工程师。
- 希望能应用敏捷方法中成熟的概念和技术来改进需求工程过程的需求工程师。
- 想要理解敏捷项目中需求工程规程学科价值和好处的敏捷专业人士。
- 希望通过使用经过验证的需求工程技术和方法来改进敏捷开发的敏捷专业人士。
- 相关学科的人员 – IT 经理、测试人员、开发人员、架构师和参与开发（大部分，但不只是软件开发）的业务的其他代表人员 – 即希望了解如何在开发过程中成功地结合需求工程和敏捷方法的人员。

细化程度

本教学大纲和学习指南的细化程度适用于国际统一的教学和考试。为了达到这一目标，教学大纲和学习指南包含了以下内容：

- 通用的教学目标，
- 教学目标的内容描述
- 参考文献的引用（如有必要）

教学目标 / 认知水平（Educational Objective / Cognitive Knowledge Level）

本教学大纲和学习指南为每个模块规定了一个认知程度等级。较高级别包含了较低级别，分别用两组动词来说明。“了解”用于 L1 级别，“掌握并且运用”用于 L2 级别。这两组动词可用下列动词替换：

本教学大纲中的所有模块和教学目标都被指定一个认知等级。使用以下级别来描述：

- **L1：了解**（描述、枚举、表征、识别、命名、牢记……）——牢记或检索以前学到的材料。
- **L2：理解**（解释、诠释、完成、总结、证明、分类、比较……）——从给定材料或情况中掌握/构建意义。
- **L3：应用**（指定、编写、设计、开发、实施……）——在给定情况下应用知识和技能。

注意，在 Ln 认知知识级别的学习目标，还包含其以下所有认知水平的元素（L1 至 Ln-1）。

示例：
学习目标为 "应用需求工程技术 xyz" 定义为 L3。然而，要达到应用的能力要求，要求学员了解需求应用工程技术 xyz （L1） 以及理解该技术的用途(L2)。

术语表中定义的所有术语都必须是了解级别（L1），即使教学目标中未明确地提及这些术语 术语表可以从 IREB 主页 https://www.ireb.org/en/downloads/#re-agile-glossary 下载。

在本教学大纲和学习指南中使用 RE 来简称需求工程（Requirements Engineering）。

教学大纲和学习指南的结构

本教学大纲和学习指南由四个主要章节组成。每个章节涵盖一个教学单元（EU）。每一章的标题都给出了学员对本章所应该掌握的认知等级，也是本章中各节的最高认知等级。此外，所建议的教学时间是学习本章课程所需投入的最少教学时间。章节中使用的重要术语列于本章的开头。

例： 第 2 章 需求工程@敏捷基础（L1）
课时： 1 ¼ 小时（1 小时 15 分钟）
术语： 产品负责人（Product Owner）、产品待办列表（Product Backlog）、冲刺待办列表（Sprint Backlog）、史诗（Epics）、用户故事（User Stories）、故事地图（Story Maps）

这个例子表明，第 2 章包含 L1 级的教学目标，需要 75 分钟来教授本章中的材料。

每章都包含不同的节。每节的标题也包含对其内容的认知等级。

教学目标（Education Objective，EO）会列在正文之前。编号显示它们属于哪一节。

例： EO 3.1.2
这个例子表明教学目标出现在 EO 3.1.2 在第 3 章第 1 节中。

考试（The Examination）

本教学大纲和学习指南是需求工程@敏捷基础级考试的基础。目前可以提供两种形式的考试：

- 受监考的正式考试，题型为选择题，通过后颁发官方认证的 RE@Agile Primer 证书，类似于 CPRE 基础级和 CPRE RE@Agile Practitionerd 的选择题考试，考试时长为 40 分钟。
- 在线形式的自我评估考试，考题为多项选择题，获得参与证明 。

监考形式的考试可以在培训结束后立即进行，但也可以与培训分开（例如：在考试中心进行）。IREB 许可的认证机构列表可在官网上找到： https://www.ireb.org/exams/bodies。

自我评估的在线模拟考试在 IREB 官网上提供。官网如下： http://www.ireb.org

考试中的一个考题可以包括本教学大纲和学习指南中的多个章节的知识点。本教学大纲和学习指南的所有章节（1-4 章）的内容均在考试范围内。

版本历史

版本	日期	备注
1.0.0	2019 年 3 月.	初始版本
1.3.0	2024 年 4 月	▪ 缺陷修复 ▪ 与 2020 年 Scrum 指南保持一致 ▪ 与 CPRE 基础级别第 3 版和 CPRE 高级 RE@Agile 第 2 版保持一致 ▪ 根据新的 IREB 定义应用的认知水平 ▪ 与新的认知水平相一致的教学目标 ▪ 调整教学单元建议教学时长，以匹配新的认知水平
1.4.0	2024 年 7 月	修复参考部分内容，应用新的设计语言

目录

需求工程@敏捷的愿景

动机和背景

不管采用哪种开发方法论，需求质量决定了整个产品开发过程的成败。与通常的认知相反，需求工程学科中的技术和方法与它们在特定开发方法论（如瀑布模型或 Scrum）的使用无关。然而，需求工程通常被视为非敏捷开发学科，这就导致了一种认知错误：需求工程知识体系和敏捷开发过程的成功没有关联。

在很多情况下，需求工程和敏捷方法是分开考虑的，而不是综合考虑的。在传统的开发过程中，需求工程是被视为系统生命周期中的一个独立学科，具有专门的角色，而在敏捷开发中，需求工程学的重要性经常被低估。

敏捷方法的基础是直接的沟通，简单的解决方案和反馈。它们的主要价值之一是针对变化的快速响应。因此，需求和优先级的变更体现了所有敏捷方法的内在概念。实际上，给予敏捷开发过程中的需求工程能力足够的重视能够促进敏捷项目的成功，同时可以持续提高所开发的系统和产品的质量。反过来说，无论应用的具体开发方法是什么，需求工程实践都可以从一些非常有用的敏捷原则和技术中收益。

目前，在很多情况下，人们要么是需求工程领域的专家，或者是特定敏捷方法应用领域的专家，因此，双方都必须找到自己的方式来利用来自对方能力领域的原则和技术带来的好处。无论是需求工程社区还是敏捷社区，目前都没有专注于两个能力领域能力的集成的认证。因此，在需求工程和敏捷方法之间，以及需求工程师和敏捷专家之间，建立一个被广泛接受的、能有效沟通的桥梁是非常有前景的。

IREB 针对这样的需求给出了答案，就是需求工程@敏捷（RE@Agile）的认证。

这样的桥梁需要从两个不同的方向来架设：一方面，需求工程社区需要理解如何成功地将丰富的技术和方法应用于敏捷开发过程，以及如何应用来自于敏捷方法的具体技术以便于改进需求工程实践。另一方面，由于敏捷方法旨在尽早交付有价值的软件，敏捷从业者需要了解如何通过应用需求工程学科中的成熟概念和技术来发挥这种效果。

需求工程@敏捷的核心原则是：

- *需求工程和敏捷方法可以互相促进*

  需求工程@敏捷可以分析需求工程和敏捷技术中可能的好处和陷阱。为此，需求工程@敏捷致力于将需求工程学科中的制品和技术应用于敏捷过程中，以及在不同开发方法的环境中将敏捷方法中的制品、角色和技术应用于需求工程过程中。
- *轻量且高度适应性的过程*

  基于需求工程@敏捷的理念，预测式和适应性的开发过程之间的区别至关重要。需求工程@敏捷提出了在敏捷开发中执行需求工程活动的一个轻量且高度自适应的方法。在需求工程@敏捷中，需求工程是一个核心学科而非一个单一的过程步骤：它是一个连续的过程 ，必须系统性地执行，并且需要高水平的技能和经验。
- *团队成员间以及团队与关键利益相关者之间的紧密协作，以及即时需求（just-in-time-requirements）*

  在所有团队成员和关键利益相关者之间的频繁沟通和紧密协作对敏捷开发过程的成功至关重要。在需求工程@敏捷中，团队与关键利益相关者一起以高度互动的方式获取、分析、细化并将需求文档化。需求工程@敏捷支持实践者在需求实施之前在正确的时间选择正确的活动以确保高质量的需求。
- *情境化和选择性的需求获取、分析、规格说明及细化*

  需求工程@敏捷基于这样的观点：并不是每个需求都需要在系统开始实施之前精确到低层次的细节上进行规格说明。相反，只有过于复杂（如：复杂到利益相关者或开发团队无法理解）或者关键（如：不能承担理解错误的风险）的需求才会被进一步细化和详细规格说明。总体过程依赖于共同的理念，即对功能性需求的变更是受欢迎的并且易于适应的。

- *避免不太相关的活动和功能并且确保最小可行产品（Minimum Viable Product – MVP）*
  敏捷原则之一就是“简洁”。根据这个原则，敏捷过程中系统或产品开发的第一阶段通常是MVP(最小可行产品）。MVP 是一个独立的、可部署的系统，该系统仅包含一些基础功能特征，给终端用户提供有仅足够得业务价值，可进行验证性学习。MVP 的最小范围可以剔除开发中的浪费，并且提供一个快速获取客户反馈的机会。下一个产品阶段通常是 MMP 阶段（Minimum Marketable Product – 最小可销售产品） – 一个具有最小功能特征的产品，能满足用户的需求，因此具有市场价值。需求工程@敏捷对下面两个非常重要的问题给出了答案，这两个问题甚至在非敏捷开发过程中也相对重要：“如何简化发布管理和产品定义流程？” 和“如何根据需求定义 MVP（最小可行产品）或 MMP（最小可销售产品）?"

关于 IREB 需求工程@敏捷认证：

CPRE 基础级中关于敏捷开发流程的知识被推荐作为前置知识。

需求工程@敏捷基础级证书面向相关领域的职业人士：项目经理、业务分析师、架构师、开发人员、测试人员以及业务相关人士。该证书注重于需求工程和敏捷专家之间的沟通，以及对这两个领域术语的理解。证书持有者可以与敏捷专家讨论需求工程，也可与需求工程专业人士讨论敏捷方法和敏捷开发。

拥有需求工程@敏捷基础级证书的人员可以：

- 熟悉需求工程和敏捷方法的相关术语
- 理解需求工程在敏捷过程中的角色和重要性以及敏捷在需求工程中的价值

1 动机和思维模式（L1）

课时： 1 ½ 小时

术语： 价值（Values）、敏捷宣言（Agile Manifesto）、实践（Practices）、活动（Activities）、冲刺（Sprint）、敏捷（Agile）

学习目标

EO 1.1.1 了解使用敏捷方法论的动机
EO 1.2.1 了解 IREB 对需求工程的目标
EO 1.2.2 了解敏捷宣言的核心价值观和相关原则
EO 1.3.1 了解原则、实践、和活动之间的区别
EO 1.3.2 了解敏捷和需求工程思维模式的区别
EO 1.3.3 了解思维模式和价值观对需求工程@敏捷的协同作用
EO 1.3.4 了解“文档”在敏捷环境上下文中的含义（与敏捷宣言中含义一致）
EO 1.4.1 了解使用需求工程@敏捷带来的好处、陷阱和误解
EO 1.4.2 了解误解的示例
EO 1.5.1 了解敏捷价值观可以转换为其他思维方式
EO 1.5.2 了解在概念性产品中使用的敏捷化方法以及示例

1.1 使用敏捷的动机（L1）

一些研究（参见 [MeMi2015]）表明，信息技术（IT）业务总体上正在经历一个本质性的变化：信息技术（IT）正在成为几个业务领域（例如：电子商务、社交媒体）和技术领域（例如：汽车或航空电子工业）的主要驱动力。因此，IT 驱动的业务领域的系统和产品必须不断更新，以适应客户或市场不断变化的需求。一旦市场发生变化，系统就必须根据这些变化进行调整。

现有的开发方法，着眼于长期的可预测性和稳定性，不适用于上述情况，往往在快速变化的业务模式或项目情况下失效。而敏捷方法由敏捷宣言（参见 1.2）驱动，已经填补了这一空白。敏捷（敏捷模式）本身是一个难以解释的术语，可以定义如下（参见 [ShYo2006]）：

> 对刺激进行的快速的整体运动并改变速度或方向。

这个定义不是来自软件工程，而是来自体育，但反映了使用敏捷方法的根本原因：如果市场或项目情况需要快速和可控的变化，敏捷方法是适合的。敏捷方法当然不仅仅是快速开发（参见 1.2），但本质上，敏捷方法的所有原则最终都侧重于按照给定的质量进行频繁的交付。这导致了频繁的反馈周期，从而能够快速响应客户的需求。

重要一点的是要意识到，敏捷方法论和敏捷模式都不会是最终目标 [Meyer2014]。一个组织必须能够选择适合其市场、客户和组织需求的开发方法。加特纳（Gartner）甚至指出，用正确的方法开发 IT 是数字化业务的主要成功因素 [MeMi2015]。

1.2 在需求工程和敏捷中的思维模式和价值观（L1）

需求工程师的心态和价值观在 IREB 的需求工程定义中进行了阐述（参见 [Glinz2022]）：采用系统化和规范化方法来规范和管理需求，目的是了解利益相关者的愿望和需求，并将交付不满足这些愿望和需求的系统的风险降至最低。

在需求工程中，我们谈论的是一个系统，而不是软件或产品。术语“系统”的使用并不意味着排除产品、其他类型的软件，甚至其他的事物（例如：业务流程或硬件）。而是为了强调系统是一组在环境中协同工作的部分或元素的事实。需求工程将环境称为系统上下文。在教学大纲和学习指南中，我们将始终使用术语“系统”，它应该包括产品和任何与软件相关的其他类型的元素。

在 IREB 的基础级中 [CPREFL2022] 还定义了需求工程的四个主要活动：获取、记录、确认/协商和需求管理。该活动列表不表示特定的一组步骤或执行这些活动的序列。IREB 基础级的核心价值是需求工程是不依赖于过程的方法：需求工程提供丰富的知识体，包括各种方法和丰富的技术集合，这些技术可以应用于任何开发方法。它不推荐或指定一个特定的过程。

这个大纲和学习指南将使用术语“敏捷方法论”来引用敏捷领域中出现的一套丰富的方法（参见 2）。为了区分敏捷方法论和其他开发方法论（如计划驱动或瀑布式），该大纲和学习指南使用术语“非敏捷方法论”。这两个不同的概念可能会被评估“哪个是最好的方法” - IREB 确信这两种方法（敏捷和非敏捷）都有其价值。

敏捷的思想体现在敏捷宣言和它的十二个基本原则内（参见[AgileMan2001]）：

敏捷宣言

我们一直在实践中探寻更好的软件开发方法，
身体力行的同时也帮助他人。

由此我们建立了如下价值观：

个体和沟通高于流程和工具。
可运行的软件高于详尽的文档。
与客户协作高于合同谈判
响应变化高于遵循计划

也就是说，尽管右项有其价值，
我们更重视左项的价值。

敏捷原则

1. 我们最优先要做的是尽早通过持续交付有价值的软件来使客户满意。
2. 即使到了开发的后期，也欢迎改变需求。敏捷过程利用变更来为客户创造竞争优势。
3. 经常性的交付可用软件，交付的间隔可以从几周到几个月，交付的时间间隔越短越好。
4. 在整个项目中，业务人员和开发人员必须每天都在一起工作。
5. 围绕着积极进取的个人来构建项目。给他们提供所需要的环境和支持，并且信任他们能完成工作。
6. 在团队内部，最有效并且最有效率的传递信息的方法，就是面对面的交谈。
7. 可用的软件是衡量进度的主要指标。
8. 敏捷过程促进可持续发展。投资方、开发者和用户应该保持一个长期的、恒定的开发速度。
9. 不断的关注优秀的技能和好的设计会增强敏捷能力。
10. 简化——最大化不必要工作量的艺术——是必不可少的。
11. 最好的架构、需求和设计出自于自组织的团队。
12. 团队应定期自省如何提高效率，并相应地调整团队的行为。

如果我们比较需求工程和敏捷模式的价值观和思维模式，彼此之间没有任何相矛盾的地方。两者价值观共有的最重要的一点，即通过满足产品最终用户的需求或者解决他们的痛点，使得他们感到满意和快乐。当然，我们也必须认识到，需求工程和敏捷模式的价值观和思维模式并不是完全相同的。它们的交集由需求工程@敏捷定义，这将在下一个教学单元中进一步解释。

1.3 将需求工程和敏捷原则联系到需求工程@敏捷（L1）

在深入讨论需求工程@敏捷的细节之前，我们必须澄清一些术语。在软件产业和软件研究中，关于开发软件时如何工作和行动已经有丰富的理论和经验。这些知识可用于多个抽象层次。下面，我们将介绍原则、实践和活动的区别，并将其作为三个抽象层次来讨论软件开发（参见 [Meyer2014]）：

- 原则是一种抽象的、可证伪的规范性陈述
- 实践/技术是一个原则在某一特定环境下的实例化
- 活动是一个实践真实的或计划好的实施

区分这三个定义的关键点是规范、抽象和可证伪。规范是控制一个活动的描述，而不是描述事件或属性。一个原则通过抽象而有别于一个实践。例如，“在交付之前测试一个软件功能”是规范，而“为每个软件特征创建单元测试”是基于给定原则的一种实践。这个例子中的一个活动可以是为一个图书馆系统的搜索功能创建一个单元测试。可证伪意味着一个有足够背景知识的人可以不同意某个原则。上述原则（“在交付之前测试一个软件功能”）满足这个标准。因为有人可能会说，测试不足以保证软件安全性特征的质量，它们应该通过数学手段来验证。不可证伪的陈述（例如“追求高质量”）不应成为指导行为的原则。

这些原则指导我们在相关活动中有意识的进行决策，我们的实践也是建立在这些原则基础之上（这本身就是一个原则）。了解哪些不同的实践遵循了哪些原则，使我们有能力根据给定的情况做出不同的反应。可证伪性促使我们讨论某一原则或实践在某一环境下的适用性，从而帮助我们决定某一原则是否应该遵循。

认识原则和实践只是第一步：正确的运用实践才是其真正的目的。例如，用例的定义是众所周知的实践，它是针对原则“重要特征的功能性需求描述”的实践。而编写一个高质量有价值的用例本身是一种能力，仅靠知道一个用例模板（样本）是远远不够的。

本质上，我们已经定义了三个层次的能力：

1. 了解原则（对应于教学目标 L1）
2. 理解遵循给定原则的实践（对应于教学目标 L2）
3. 实施某一领域的实践（实施高质量活动的能力）（对应于教学目标 L3）

比较敏捷模式与需求工程的行为准则（参见 1.2），有可能看到为什么这两种学科有时被认为是冲突的：需求工程关注的是系统地获取需求，并文档化需求作为制品进而发挥它们的作用；而敏捷强调可用软件的重要性超过全面的文件，并且认为个人以及相互间的沟通价值高于过程和工具。

在实践中，对于这两种思维方式的过度解读会导致冲突：其中关于需求工程的一个错误解释是，可以创建一个完整、一致并且各方均认可的需求文档，并且在实现需求的过程中不需要进行进一步的修改。同样在敏捷模式中一个类似的错误解释是，项目开发可以在没有任何准备工作的情况下开始

，并且只通过持续发布软件，定期由利益相关者检查软件[注意：从需求工程的角度来看客户是利益相关者的一个子集。]，并通过反馈进行改进就可以获得成功。

我们可以确定需求工程和敏捷模式实际上并没有冲突：两种方法都有相同的目标，即以明确定义的质量水平交付软件。敏捷方法可以让项目组高效快捷地提供可用软件（缩短周期时间）。需求工程提供了适当的技术来了解利益相关者的愿望和需求，从而指导项目组开发出正确的软件。

需求工程可以帮助促进以下内容：

- 为开发有价值的软件而去理解用户的愿望和需求（第一敏捷原则）
- 用适当的工具识别市场的变化，为利益相关者创造竞争优势（第二敏捷原则）
- 选择适当的工具和技术促进利益相关者和开发者之间的有效协作（第四敏捷原则）
- 用适当的工具和技术来支持口头交流（第六敏捷原则）
- 为减少不必要软件的开发而去理解用户的愿望和需求。（第十敏捷原则）

需求工程在敏捷模式和其他开发方法中应用的重要区别在于应用的时间和过程。在这个教学大纲和学习指南中，国际需求工程委员会定义了需求工程@敏捷，它展示了如何在敏捷方法论的环境中应用需求工程。

与“敏捷需求工程”相比，国际需求工程委员会更喜欢需求工程@敏捷这一术语，以明确需求工程是独立于过程的。

敏捷软件开发宣言强调了持续开发的可用软件（或制品）价值高于详尽的文档。极端的解释会造成对敏捷方法的错误印象，即敏捷模式完全放弃了文档。这个解释是错误的：当有目的性文档能够支持开发或是产品一部分的时候，它们在敏捷模式中仍然受到欢迎和推荐。过去的开发过程中一个明显的问题是，许多项目没有明确的目的或附加价值就创建了文档，而这种文档应根据原则尽量避免。

1.4 使用需求工程@敏捷的好处、误解和陷阱（L1）

需求工程@敏捷将提供诸多好处。然而，这些好处也是有代价的：人们应该避免对其产生误解和错误使用。

1.4.1 需求工程@敏捷带来的好处

在同一团队中具备需求工程&开发能力可以减少交接：敏捷方法中跨职能团队的实践要求团队成员具备能完成基于选定需求的增量开发产品的所有技能。

在团队中执行需求工程任务可以减少预先创建详尽需求文档的需要，因为团队成员可以直接向其他成员解释某些细节。在这种情况下，文档的作用将是记录讨论结果和保存知识。

增量开发可以优化现有的想法：敏捷方法的核心原则是在迭代中完成软件增量开发。迭代过程创造制品（例如，业务流程模型、史诗、用户故事、用例、用户界面原型、过程描述或软件），并在一

系列的开发和评审活动中改进这些制品。此过程的好处是制品和/或软件的质量被不断地改进和优化。此外，较小的增量允许较早与客户讨论，并降低客户期望和开发成果之间存在巨大差距的风险。

细化是一个用于完善和确认需求的原则：在敏捷开发中，已经开发了一个非常好的实践——持续细化。在这个实践中，所有利益相关者密切沟通，定期召开会议，在当前的基础上为开发团队评审和细化需求。此外，“准备就绪”（Definition of Ready，DoR）这个定义作为一个好的实践被用作品质关，来确认需求是否已经准备好在后续的迭代中去实施。

需求工程帮助定义初始的产品待办列表：需求工程提供了不同的技术，以获得利益相关者对所需产品的需求的正确理解。因此，需求工程提供了对初始定义的产品待办列表所限定的需求更深入的理解。重要的是认识到这样的需求工程活动并没有创建详细的规格说明。相反，这些活动的目标是专注于对某个抽象层次的产品的彻底理解（例如，理解和定义必要的用例或史诗和用户故事）。例如，一个复杂的高级别的业务流程可以使用确定的需求工程方法分解成史诗、功能以及用户故事来创建最初的产品待办列表 [CPREALAGILE2022]。

1.4.2 需求工程@敏捷的误区

在开发的世界（主要是软件开发）中，存在一些与需求工程有关的误区和陷阱，我们将展开讨论：

错误观念 - 需求工程只是前期的分析：通常，需求工程被认为只是一种前期活动。实际上需求工程作为一门学科，是独立于过程的，并不强制完成完整的前期工作。相反，在敏捷方法中需求工程的活动可以与其他活动（例如编码或测试）一样执行。需求工程是一个嵌入在每个迭代中的活动。

错误观念 - 事先（活动）是不好的：一个迭代开发的准备活动是所有非日常开发的重要组成部分。事先思考并不意味着自动生成一个特定的软件生命周期或冗长的分析阶段和繁琐的文档：根据项目的实际情况确定应该如何以及何时去事前思考。用户不应该教条地将敏捷的专业文献理解成事前思考是件坏事；宣言和上述原则都不支持这种理解。关注事先思考价值的敏捷实践有（用户）故事映射（Story Mapping）（参见 [Patt2014]）、原型化（参见 [Martin1991]）和测试驱动开发（TDD）（参见 [Beck2003]）。

错误观念 - 需求工程等同于文档：需求工程经常只与它产生的相关文档关联起来。然而，文档仅仅只是创造知识的活动的一个可能结果。好的需求工程能意识到，即使是最好的文档也不可能是完整的。相反，一个文档的作用是用来支持目标，如在 3.2 所定义的目标：符合法律要求（遵守）、保存有价值的信息、简化沟通和支持思维过程。

错误观念 - 有用户故事就够了：用户故事是一种获取利益相关者需求的常用方法。然而，此方法仅仅只是启动沟通，不是描述完整的规范说明。敏捷实践中的“3C-Card/卡片、Conversation/对话、Confirmation/确认”总结了如何从一个粗略的需求通向完整需求。通过将用户故事与其他认可的需求工程技术（如环境图、原型设计、用例和用户旅程）相结合，可以实现更全面的需求描述。

错误观念 – 文档是没有价值的，只有代码才有持久的价值：虽然在一些过程繁重的项目，过度规范可能是个问题，但是断定所有的文档是没有价值的是不对的。需求文档，就像设计文档、测试文档或操作文档一样，在特定环境下有其存在意义。这些文档都是任何可持续软件产品开发过程中产生的同等有效和必要的制品。

错误观念 – 可用软件是确认需求的唯一方法：敏捷宣言重视“可用软件而不是全面的文档”。当涉及到确认需求时，从以上敏捷宣言中可能得出的错误结论是，用可用软件来确认需求总是优于用文档形式的需求来确认。如果通过软件确认需求的方法所耗费的成本和承担的风险与结果相比是可接受的，那么软件作为确认需求的手段是可优先考虑的。如果成本和/或风险很高，需求工程提供若干工具，允许项目组在编写一行代码之前快速反馈和确认需求，例如：用户界面模型或故事板。

1.4.3 需求工程@敏捷的陷阱

陷阱 – 将需求视为一种固定类型的信息：在所有需求工程实现的环境/方法中，常见的典型错误是将“需求”视为固定类型的信息。由于这种误解，导致编写需求文档通常被认为是浪费时间，因为需求会改变得如此之快以至于它们一写下来就失效。需求不是一种固定的信息类型；需求可以通过不同层次的细节、抽象和格式表述。例如，系统的愿景或目标是高抽象级别的需求，通常具有较长的生命周期；可执行原型是确认一组需求或获取新需求的手段。

陷阱 – 失去大局观：敏捷方法经常会被误解，在实施过程中只关注团队当前的主题。从开发者的角度来看，这可能被认为是一个有用的原则，因为开发者可以把精力集中在手头的工作上，而不会被长期目标分散注意力。然而，如果每个人只关注手头的工作，就会失去全局观和远见。可持续敏捷方法论通过各种专题会议处理长期和中期的目标（例如，细化专题会议、指导方针专题会议或总体战略研讨会）。与本陷阱相关的另一陷阱是项目组在没有彻底清晰定义业务问题（通常被称为需求）的情况下，已经开始推进解决方案。

陷阱 – 利益相关者信息过载：在敏捷团队中，需求工程的制品可以包含高密度的信息，并且会非常快速地被创建。在一些“前沿”或高技术项目，如果这些领域很难找到技术专家时，会经常出现这种情况。这会导致在整个迭代过程中都在处理这些需求工程的文档，以至于利益相关者消耗大量评审时间，因为他们必须面对详细的规格说明。适当的综合规格说明和开发（原型）也许是更好的处理方式。

陷阱 – 以增量和迭代的方式处理每一个主题：一个系统中不是每个需求主题都应该以细粒度和增量的方式开发。增量开发特别适合于复杂的需求主题（参见[Meyer2014]）。一个这样需求主题的典型案例是可以划分成相互独立元素的过程（例如，一个在线商店中的购买过程）。更高复杂度的需求主题（参见[Meyer2014]）不适合增量开发，因为每一个关于一个主题的新认识都将导致对已知信息的全新理解。例如有复杂的输入参数和简单的输出参数的计算过程（如保险策略或发动机控制组件）。

陷阱 – 增量开发可能不鼓励激进的或破坏性的创新：敏捷增量开发过程不鼓励创新和/或破坏性的想法，因为给定的制品（例如软件或软件的一个特征/功能）一旦完成定义，通常仅进行局部改进（例如，修正错误或添加缺失元素）。虽然敏捷宣言明确地欢迎变革，但敏捷的增量开发过程通常更支持产品和服务的持续创新， 如果要考虑不同的想法并且重新组合现有的想法，才会出现激进的或破坏性的创新[LiOg2011]。就软件而言，在敏捷环境中开发可选的方案通常被认为是浪费（参见第十原则——减少不必要的工作）。对于激进的或破坏性的创新，需要引入额外的实践，如 1.5 所提出的精益创业思想或设计思维方法。

主要的和最严重的陷阱 – 敏捷和文化的变迁不一致：敏捷价值观促进组织工作方式的改变，失去了对某些可交付成果的所有权和及降低了集体责任感。此外，敏捷要求团队对自身行为持续回顾和总结，以改善其工作方式：一个团队，有时整个公司会不可避免的遭遇反复变化， 这种文化（组织层）的变化不仅需要时间，也需要优秀的人员才能使团队朝着一个新的方向前进。

1.5 需求工程@敏捷和概念框架 (L1)

为了应对由软件工程领域外引发的挑战，在软件工程界出现了敏捷开发（参见 1.1）。然而，这些挑战并不仅仅是软件工程界所经历的， 工业和社会的其他领域也面临着类似的挑战，比如挑剔的客户和更快的创新周期。其他领域开发了与敏捷开发非常类似的概念框架（即创建概念或系统规格说明）。其中一些方法从需求工程的角度来看特别适合于开发创新和产品愿景。在此将简要介绍它们，包括讨论它们与敏捷思维的对应关系（参见 1.2）。在下文中，我们将介绍三种方法作为敏捷概念框架的示例。

设计思维（Design Thinking）（参见[Dsch2015]，[LiOg2011]）是一种解决所谓“棘手问题/Wicked Problems”的方法。从需求工程的角度来看，设计思维是获取和确认技术的结合。该方法的核心是（a）一个多学科的团队，该团队致力于解决问题并代表了解决问题所需的广泛知识；（b）一个工作环境，团队可以在其中共同开发想法；以及（c）一个迭代过程，由以下阶段组成：

- 共情：在这个阶段，团队沉浸在与遇到问题的当事人同样的感受中，对需要被解决的问题产生理解。
- 定义：在这个阶段，团队对所要解决的问题进行重新表述，以便对问题的细节有一个共同的理解。
- 构思：在这个阶段，团队专注于构思解决方案。这里的目标不是产生某个特定的解决方案。相反，团队会构思出尽可能多的想法。当这个阶段结束时，团队将选择最有希望的原型设计方案。
- 原型：在这个阶段，团队根据确定的想法创建非常简单的原型（不一定是软件！）。这里的原则是原型应该尽可能真实，并且尽可能降低成本。
- 测试：在这个阶段，团队和真实的客户一起测试原型，以获得对他们想法的反馈。测试阶段的一个主要原则是“展示而不是说明”，即原型应该能够为自己说话，以便用户可以提供真正的反馈。

设计思维的阶段是可扩展的，并且适用于从几天到几周的项目。此外，没有预先严格定义通过阶段的顺序。只要有必要，团队就可以决定在这个过程中往回走或者向前走。从这一点来讲设计思维符合敏捷价值“响应改变高于遵循计划”。设计思维过程的最终结果是一组原型，它包括了对最初定义的问题的确认和创新的解决方案。因此，设计思维支持第一敏捷原则，可以用来开发高价值软件的构思。如上面几段所述，综合学科团队和工作环境是设计思维过程的核心要素，这与第五敏捷原则相一致。而原型阶段的主要目标是产生成本低且轻量级的原型，这一点也符合简单化的敏捷原则。

设计冲刺（Design Sprint） [KnZK2016] 是一个为期五天的过程，这个过程中团队遵循与最终用户一起设计、原型化，以及和最终用户一起测试的原则来开发解决方案。从需求工程的角度来看，设计冲刺也是获取和确认技术的结合。这种方法的核心是时间盒式的工作方式，每天团队都致力于下列活动之一：分析提取团队所知道的信息、起草解决方案、选择要原型化的方案、原型化已选定方案、并最终与真正的用户一起测试这些方案。设计冲刺与敏捷开发之间主要的区别在于给出的原型不一定是软件。此外，重要的一点是要认识到“冲刺（Sprint）”这个词不是指“Scrum Sprint”。

精益创业（Lean Startup） [Ries2011] 是一种适用于开发业务和管理初创企业的方法，在敏捷社区中很受欢迎。从需求工程的角度来看，它也包含了一些非常有趣的想法。其中两个例子是特殊产品开发方法和最小化可用产品。产品开发方法被称为构建-量化-学习，特别关注持续的学习客户需求。最小可用产品（Minimum Viable Product-MVP）是“一种新产品的版本，它允许团队以最少的工作量收集尽可能多的关于客户的经过确认的信息” [Ries2011]。精益创业的另一个重要概念是结构性的改变，“一种结构化的路线修正方法，该方法用于测试新的假设是否成立，这些假设是关于产品、战略和发展动力的” [Ries2011]。从需求工程的角度来看，这些想法是获取和确认技术的结合。不是基于概念或文档来获取和确认需求，而是用真实的产品来执行获取和确认。根据 Ries 的观点，在极端不确定的情况下后者是更好的选择。Ries 强调 MVP 不一定是具备全部功能和完整的软件。相反，这本书提到了一个例子，有一个非常简单的售鞋网站，这个网站通过人工发货过程来确认网上的购鞋需求。

这些方法表明敏捷不仅仅是 Scrum。设计思维、设计冲刺和精益创业的例子表明，在需求工程@敏捷中有一些思维活动的方法，它们共享敏捷方法的思维方式，因此需求工程@敏捷与想要以敏捷方式开发软件的组织完全兼容。所有这些方法应被视为瀑布式方式的一种新形式，或在事先考虑中不可忽略。它们可以并且应该在敏捷开发的框架内（例如在一次或多次迭代中）被使用，来设计系统的特定方面。或者，它们可以用作敏捷开发之前的活动（例如，前置思维的快速阶段）。由此可见，这些方法有助于提升在敏捷开发中潜在的创新能力。

2 需求工程@敏捷的基本原理（L2）

持续时间：2.5 小时

术语： 产品负责人（Product Owner）、产品待办列表（Product Backlog）、冲刺待办列表（Sprint Backlog）、史诗（Epics）、用户故事（User Stories）、故事地图（Story Maps）

学习目标

EO 2.1.1 了解敏捷方法论的例子
EO 2.2.1 Scrum 是敏捷的一个例子，了解 Scrum：Scrum 的角色、流程、制品以及它们与需求工程的相关性
EO 2.2.2 了解 Scrum 产品负责人的职责
EO 2.2.3 了解产品待办项的概念及其与需求规格说明文件的区别
EO 2.3.1 理解典型需求工程师和 Scrum 产品负责人之间的区别
EO 2.4.1 理解为什么需求工程应是持续过程的一部分
EO 2.5.1 了解价值驱动开发（即需求的优先级排序）
EO 2.5.2 了解价值是风险与机遇管理
EO 2.5.3 了解赢利组织和非赢利组织中价值的实例
EO 2.6.1 了解如何简化产品定义流程以及如何定义最小可行产品
EO 2.7.1 了解持续过程的价值和它们的学习曲线

2.1 敏捷方法论 （概览） （L1）

目前已开发出许多遵循敏捷宣言价值观的方法论。本教学单元将会为您提供其中一些方法的概览，以便了解方法论的多样性。当然这里不会详尽地列出所有的方法论，而主要是会从需求工程的角度讨论这些方法论。

水晶方法论（Crystal）是由 Alistair Cockburn [Cock1998] 建立的一系列方法。他提议每个项目需要有其自己量身定制的过程模型。基于项目的规模、复杂程度和关键程度，Alistair 建议使用适合的角色、活动和制品。Crystal Clear 透明水晶方法论适用于规模较小并且不太重要的项目，该方法与 XP（极限编程）非常相似。而橙色水晶方法论（Crystal Orange）和红色水晶方法论（Crystal Red）则增加了一些形式以应对更大规模的项目。从需求角度来看，Alistair 建议（除其他事项外）可以使用低仪式感的用例模型和仿真模型。

精益开发（Lean Development） [Popp2003] **和看板（Kanban）**是基于 20 世纪 40 年代首次用于汽车生产的原则。这些方法已经被改进并适用于敏捷方法环境中的 IT 项目[Ande2012]。精益开发和看板方法试图发现生产过程中的 7 种浪费（未完成的中间产品、生产过剩、缺陷等）并逐步消除这些浪费从而加速最终交付。

Scrum [Scrum2020] 是一个框架，该框架用于在复杂环境中将有价值的解决方案作为产品进行开发和维护。该框架专注于基于经验及精益思维的迭代和增量开发。该框架仅定义了三个关键角色（Scrum [Scrum2020]中称为职责）:产品负责人（Product Owner，负责管理产品待办列表，即定义产品的愿景和所有相关需求）；开发人员在冲刺期（Sprint，Scrum 的迭代周期）内实现这些需求；敏捷专家（Scrum Master）指导和促进 Scrum 适用于 Scrum 团队和开发组织。我们将在下个章节里更加详细地介绍 Scrum。

测试驱动开发 TDD（Test driven development） [Beck2003] 是基于对相关特性进行代码开发之前先编写好测试的理念。测试用例是对产品所要实现需求的准确和详细的规格说明。

极限编程 （eXtreme programming – XP） [Beck2004]：强调用户和程序员进行直接沟通（即“现场客户”就坐在程序员身旁，对需求不断进行商讨，并以实现特征的形式获得即时反馈）。

2.2 以 Scrum（以及 Scrum 的最佳实践）为例子（L1）

Scrum 是最流行和最常用的敏捷框架。Scrum 是一个轻量级的框架，该框架通过帮助个人、团队和组织通过对复杂问题的适应性解决方案来创造价值。Scrum 指南 [Scrum2020] 提供了 Scrum 及其基本组件的定义。

根据这种方法下，通过定义 [Scrum2020]， 可以有:

- 三个职责/角色（敏捷专家、产品负责人、开发团队）。这三个角色也被统称为 Scrum 团队。
- Scrum 的五个核心活动：冲刺 (Sprint)、冲刺计划 (Sprint Planning)、每日站会 (Daily Scrum)、冲刺评审 (Sprint Review)、冲刺回顾 (Sprint Retrospective)。
- 三个制品[1] 以及他们相关的承诺(产品待办项、冲刺待办项、增量)。

Scrum 并不推荐任何工程实践活动。

Scrum 建议通过一系列的冲刺，每次迭代在一个月或更短的时间盒内，对产品进行迭代和增量式的开发。每个冲刺都交付一个可用增量，它是实现产品目标的实在的垫脚石。在此，产品负责人决定是否将增量交付给客户；通常情况下是会交付的，除非存在重大原因（例如潜在风险）。

[1] 符合 CPRE 术语的制品 [Glinz2022]

冲刺

冲刺是开发过程的主要驱动环节，因为它是一个遵循 PDCA（计划-执行-检查-处理）的流程，允许较短反馈周期的迭代过程。每个*冲刺*开始于*冲刺计划*，在该计划上，*Scrum 团队*成员一起共同协作定义需要在下一个*产品增量*交付哪些功能以及如何实现（分解）。该计划源于从产品待办列表里拉取的待办项（产品待办列表是一个有序、动态的包含产品所有当前已知需求的列表）。与传统需求工程中的需求规格说明文档相反，产品待办项还不是所有需求的完全规格化的集合。通常（尤其是在产品开发之初），它只是一个粗略制定的想法列表，这些想法随着时间的推移而发展并成为具体的需求。产品待办项的元素优先级越高，离实现方法的时间越近（例如，在接下来的两个 Sprint 中的一个），对产品待办项元素的描述就越精确。产品负责人负责产品待办项列表，列表中待办项根据实际产品目标来引导。具体的请看 3.1.1 规格说明文档与产品待办列表。

*冲刺计划（Sprint Planning）*用来选择将在下一个冲刺中实施的元素（产品需求）。它会产出三个关键结果：冲刺目标（Sprint Goal）（做什么）、冲刺待办事项列表（Sprint Backlog）（做什么内容，即选中需求的细分和拆解），以及冲刺计划（Sprint Plan）（如何做）。

随后，*Scrum 团队*便开始开发*产品增量*。*开发人员*负责将选定的项目转化为可交付成果。开发团队与*产品负责人*一起优化*产品待办项*，并分享当前进度的反馈。开发团队每天通过每日站会同步工作进展，评估冲刺目标的推进情况，并根据需要更新冲刺待办事项列表。

*开发团队*的工作是在“已完成的定义”（Defination of Done- DoR，对于增量需要实现完成的定义）的指导下进行的。“已完成的定义”可以帮助利益相关者清楚地了解产品进度。

当冲刺时间框架结束（最长不超过 1 个月）后，*Scrum 团队*和关键利益相关者会在*冲刺评审（Sprint Review）*中检查产品增量（Product Increment），以评估冲刺结果并讨论下一步计划。产品待办列表会进行相应更新。

冲刺结束于*冲刺回顾会议*，在冲刺回顾会议上 Scrum 团队会对如何提升效率、哪些做得好、哪些需要改进进行审查。*冲刺回顾（Sprint Retrospective）*一结束，即标志着上一个冲刺的完结，紧接着新的冲刺就会开始。

最佳实践

尽管 Scrum 并非起源于需求工程技术，但敏捷社区却开发出了一些适合 Scrum 的最佳实践。

Scrum 指南把产品待办事项列表中的每个条目称为 "产品待办事项"。是一个用来识别待开发产品所有信息的通用词语，但许多产品待办列表项其实就是需求或者通过优化即可成为需求。

敏捷社区建议：

- 将需求分解成：史诗、供选的特征、用户故事（粒度水平），参见 [CPREALAGILE2022]。
- 将功能性需求（能力）、质量需求（行为）以及约束（环境）区分开
- 按照主题将需求进行分组

一个（额外的）最佳实践被开发用来描述产品待办列表的特点：即待办列表应当是“DEEP”的（Detailed appropriately-详略得当的，Estimated-可评估的，Emergent-随时发生的，Prioritized-有优先级的） [Cohn2004]。在冲刺阶段内，需要对待办列表进行梳理和优化，这也是敏捷开发过程中的核心部分。

通过“完成的定义” （DoD），团队建立了共同的理解，用于判断产品待办事项何时完成并可以纳入可用增量中[Scrum2020]。

另一个最佳实践则是运用 INVEST 原则[Wake2003]。其首字母包括以下准则：

- **I**：（Independent of each other）（用户故事是）彼此相互独立的
- **N**：（Negotiable）（用户故事是）可以协商的
- **V**：（Valuable）（用户故事是）对用户有价值的
- **E**：（Estimable）（用户故事是）可评估的
- **S**：（Small）（用户故事）足够小，可以在一个冲刺内完成
- **T**：（Testable）（用户故事是）可以测试的

需求工程中也提出好需求的相应标准（参见[CPREFL2022]）：

- 确认的
- 无歧义的
- 必要的
- 一致的
- 可验证的
- 可实现的
- 可追踪的
- 完整的
- 易于理解的

2.3 需求工程师与产品负责人之间的区别和共同点（L2）

在讨论了 Scrum 框架的原则后，让我们来对比下传统需求工程师角色与产品负责人角色。必须提到的是，产品负责人的角色同时也被许多敏捷方法所采用（除了 Scrum 之外）。因此，以下描述不一定只适用于 Scrum。

需求工程师的核心活动包括 [CPREFL2022]：

- 需求获取
- 需求记录
- 需求的确认与协商
- 需求管理

如上所述，产品负责人的主要责任包括：

- 确保团队持续交付（业务）价值：这意味着产品负责人需要平衡产品远期愿景（产品目标、产品愿景）和短期需求，根据定义的标准对产品待办事项列表进行优先排序，并在每个迭代（冲刺）结束时与利益相关者一起检查团队的结果。
- 管理所有利益相关者：产品负责人负责将一致的需求传达给团队。他需要从所有利益相关者那里收集需求并确保它们相互之间不会有冲突。他需要解决利益相关者之间所有的冲突，以使开发团队免于此类冲突。
- 持续为开发团队提供产品待办列表里具有最高（商业）价值的事项。这些需求的粒度必须要足够小，能够在一个冲刺内完成。对于在冲刺计划会议结束后出现的所有问题，产品负责人必须能快速做出澄清。

通过对比这两个角色，不难发现需求工程师和产品负责人都必须要（与所有其他利益相关者一起）执行获取需求，编写需求文档，确认需求和管理需求的关键任务。只是在敏捷环境里，所使用的标记符号和工具相对没有那么正式：

- 例如，故事卡片而非需求文档
- 更多的交谈，较少的书写
- 更多强调需求当前的状态，较少强调版本控制和历史

在持续对高质量需求全面负责的时，产品负责人的角色比传统需求工程师更加广泛，因为他/她需要对产品整体的成功进行负责，并能持续地收集业务反馈并对待办列表进行相应的更新和进行优先级排序。

就需求工程活动而言，我们可以总结为产品负责人对其负责。他可以得到具有需求工程经验的人员的支持，也可以自己执行这些任务。然而，产品负责人不仅要对需求工程的结果负责，同时也要对产品待办事项列表负责。

2.4 需求工程作为持续过程（L2）

在敏捷里，需求工程在开发过程中是相对不太明显的阶段，而更多的是一个迭代与持续性的活动。在设计与实施之前的获取和分析所有需求不是目标；而更加注重创建需求和产品的迭代和增量。

因此需求工程是一个与产品开发一起持续进行的活动。但是这个过程具有明确定义的中间结果：保证最大（业务）价值的预测性需求应当是根据定义的标准，“准备好实施”的（从上述描述的“准备就绪”的意义上来看）。只有当完成了紧急的需求后，其他在业务角度相对不紧急的需求才能开始进行优化。

“持续性过程”并不排除一些重要的前期活动。即使需求已经在基于“需要知道”的基础上得到了澄清解释，但需求的一些方面仍然需要在生命周期早期得到解决， 比如像定义愿景与目标、了解利益相关者以及建立产品范围。如果没有这些活动便开始实施则会增加风险等级。

2.5 价值驱动开发（L1）

敏捷方法力图持续地为最终用户交付（商业）价值。通常商业价值可以直接表述为财务术语，增加的市场份额，或提高客户满意度。该方法经常用于盈利驱动的组织内，但是对于非盈利组织就相对不太容易清晰地定义价值。在这点上，像与产品相关的使用率或满意指数这样的衡量指标（例如：网站的点击率或者非盈利组织的捐赠数量）或许更为相关。

另外一种价值则是降低风险。好的敏捷方法会在迭代过程中尽量在商业价值与降低风险之间进行平衡。

为了确定哪些需求会带来更高价值，敏捷方法通常会采用最小可行产品（MVP）或最小可销售产品（MMP）的方法，如下一节所述。

2.6 以简洁为本（L1）

在复杂的世界里，简洁是一种通过以下过程来接受复杂性的方法：

- 为问题创建简单以及可能不完整的解决方案，从而创建一个小的价值增量。
- 根据实践经验，获得更多了解上下文（环境）的能力。
- 以可持续的速度调整和迭代价值的创造和交付。
- 通过快速失败和快速学习从非盈利的想法中节省出资源然后重新分配至新想法中。

简洁在某种情况下与“完美”是相反的。需要说的是，简洁大多时候并不意味着“质量差”，反而是一直高质量的“最小范围”或“最小服务”。质量是不可妥协的。

敏捷宣言将“简单”描述为“最大化未完成工作”的艺术。（敏捷提倡 ‘少即是多’，只做真正重要的事情）。这句话并不是鼓励为了减轻工作量而逃避努力，而是强调明确需要完成的工作，创造价值，服务于特定目标。如果团队实现与目标不一致的需求，那就等于最大化了无价值的工作，而这正是应该避免的，以免无缘无故地浪费成本和时间。

通常定义了两类“简洁”的产品：最小可行产品和最小可销售产品

最小可行性产品（Minimum Viable Product-MVP）是一个起源于精益创业的概念（参见[Ries2011]），它定义为可以创建终端用户体验并为团队提供反馈的最小产品。这种反馈是产品进化的主要输入。许多初创公司正是以这种方式进行调整，因为它能够在低风险水平下创造快速的投资回报。

最小可销售产品（Minimum Marketable Products-MMP）的目的则更进一步。问题不仅是提供尽早的反馈以推动后续的需求活动，而且会即刻创造价值。许多的产品在不具备所有期望的功能和质量时就可以在简化“版本 1”中使用，并创造收入。这些收入再投入到产品进行持续改进。在此，精益创业过程（构建、测量、学习）经常被反复使用。

2.7 审查和调整（L1）

许多敏捷方法强调频繁与快速反馈的重要性。团队会在每次迭代后（有时甚至更频繁）讨论开发过程是否适合他们，或者是否需要改进。

在反馈过程中，每位成员都应该在早期对当前的开发过程进行审查。团队需要质疑所使用的方法论、工具以及团队的协作等。每位成员都被要求回答这样的问题：有什么运行的还不错？有什么运行的还不够好？在下一个迭代里应该进行什么尝试?

在每个迭代冲刺结束时对产品或解决方案进行评审是很重要的。团队应该检查他们的速度，然后对下一个迭代的计划实施速度进行改变或调整。

这也同样适用于需求过程。以上的这些观察分析结果应该用于过程改进步骤的短期调整。

3 需求工程@敏捷中的制品和技术(L2)

持续时间：2.5 小时

术语： 产品愿景（Product Vision）、产品路线图（Product Roadmap）、产品待办列表（Product Backlog）、冲刺待办列表（Sprint Backlog）、用户故事（User Story）、验收标准（Acceptance Criteria）、特征（Feature）、功能性需求（Functional Requirement）、质量需求（Quality Requirement）、史诗（Epic）、环境模型（Context Model）、故事地图（Story Map）、就绪定义（Definition of Ready）、已完成的定义（Definition of Done）

学习目标

EO 3.1.1 了解传统规格说明书与待办列表之间的区别
EO 3.1.2 了解愿景与目标的价值
EO 3.1.3 了解在需求工程中使用环境模型的附加价值
EO 3.1.4 了解 如何 区分 三种类型的需求
EO·3.1.5 理解需求的不同粒度级别
EO 3.1.6 理解在敏捷过程中针对不同制品类型的不同规范说明格式（例如文本格式、基于模板的格式、图形格式）
EO 3.1.7 理解术语、术语表和信息模型的价值
EO 3.1.8 理解敏捷需求工程过程中质量需求和限制条件的规格说明
EO 3.1.9 了解验收和适配准则
EO 3.1.10 理解敏捷需求工程过程中就绪的定义和完成的定义
EO 3.1.11 了解原型、增量及探针的区别
EO 3.1.12 了解在需求工程@敏捷的过程中，不同类型的制品（环境模型、史诗、用户故事、待办项、路线图、需求、完成的定义、就绪的定义）
EO 3.2.1 理解在敏捷需求工程中如何获取需求
EO 3.2.2 了解在敏捷需求工程中如何创建和维护待办列表。
EO 3.2.3 了解在敏捷需求工程中如何进行需求确认和需求协商
EO 3.2.4 了解如何在需求工程@敏捷中管理需求

3.1 需求工程@敏捷中的制品(L2)

3.1.1 规格说明文档与产品待办列表

为了根据需求创建产品或解决方案，需求不能单独存在，而是需要组织成一个具有优先级顺序的列表并文档化，而且应该完整包含可能在产品中需要的所有内容（ [Scrum2020]）。无论采用何种方法论，该需求收集过程都被视为需求工程师、业务分析师、产品负责人或任何负责需求分析的人员的主要工作成果。在不同的方法论中对这些需求集合会使用不同的形式和名称。

在需求工程中通常将这些称为制品、获取过程的结果、用户需求规格说明、系统需求规格说明或软件需求规格说明，这取决于谁收集和整理这些需求集合以及其粒度。它不一定是基于文档的，可以承载于任何物理形式（纸张、储存库、数据库……）的需求集合。

敏捷方法，特别是 Scrum，将产品待办事项列表定义为未来要实施的所有需求（以及其他产品相关信息，参见 2.1）的集合，而冲刺待办事项列表则包含已选定用于下一个迭代（Scrum 中称为冲刺）的需求[Scrum2020]。此外，这些待办列表的物理形式也不重要。待办项可能包括索引卡或墙上的便签，或者被记录在一些适当的软件工具中。尽管名称和处理方式可能有所不同，但所有制品都遵循相同的思想，并为需求的获取、记录、协商、确认和管理提供了基础。

在 2.2 中，我们了解到，DEEP 是一种很好的做法，产品待办项应该是适度详细的（Detailed appropriately），可评估的（Estimated），随时发生的（Emergent），设定了优先级的（Prioritized）。这些特性对于产品待办项非常重要，它们相互关联或依赖。对于产品负责人这个“价值优化者”而言，产品待办事项列表的排序和优先级是实现最大价值和最重要的任务。评估支持这一点，适当的细节有助于关注重要部分，因此评估及细节对排序提供支持。与记录需求的方式无关，某些元素肯定应该被捕获。这包括各种需求类型：目标和愿景、系统或产品边界的定义、功能性需求、质量需求和限制条件以及术语表（即相关术语和缩写的定义）。后文将会讨论到，在这些需求规格说明的符号、句法和细节等级上，方法可能有所不同。没有任何规格说明（例如，在没有任何书面需求的情况下，只相信利益相关者之间的口头交流）通常不是一种可选的办法，因为书面文档通常是协商、验收测试、法律要求等的基础。所有利益相关者之间的交流越多，记录的必要性就越少，但是作为结果，需求仍然应该以一种众所周知和可信的形式记录（写或画）下来。在接下来的段落中，将详细讨论这个总体需求集合的不同部分。

3.1.2 愿景和目标

每个开发过程都应该由定义产品功能的愿景或目标来导向，如果相关的产品功能实现了，则该解决方案被认为是成功的。尽早将这些愿景或目标与所有的利益相关者达成一致，对于系统或产品需求相关的所有活动都是至关重要的。

在敏捷开发的过程中，“产品愿景”这个术语通常被用来强调开发过程的每一个产出都应该有一个与产品愿景相关的独特的价值。为了定义这种价值，可能有必要首先明确定义一家公司所追求的价值。

来自不同利益相关者的目标可能是相互矛盾的，这意味着利益相关者必须通过谈判达成一致的愿景或目标。或者，来自不同利益相关者相互矛盾的目标可以用产品的变体来解决（例如一个小尺寸的 iPhone 和大尺寸的 iPhone），甚至可以定义完全不同的产品来实现（iPhone 和 iPad）。

敏捷开发通常使用不同粒度的目标，例如不同时间界限或计划间隔的目标。例如，交付物（时间和内容）的协商可以是一个年度目标；发布计划以及下一个迭代/冲刺的迭代/冲刺目标可以是一个三个月的目标 [High2009]。

产品长期的愿景和短期的冲刺目标对于强调在一个特定的时间范围内应该达到的最重要的成就是有用的，并且有助于使所有的利益相关者在一个“共同任务”上保持一致。所有这些目标都可以在路线图的时间轴上有效地表示。

产品愿景或目标是最抽象的需求形式，在没有进一步细化的情况下不能进行开发。它们为整个敏捷开发过程提供了易于理解和全面的指导。每个需求都应该检查和目标之间的关联，以验证它们对不同目标的贡献。一个与目标没有建立关联的需求，可能意味着这个需求缺乏价值。因此，产品愿景的陈述和利益相关者所达成一致的目标对于敏捷开发过程的成功是非常重要的制品，因为它们既为所有开发活动设置了框架，同时又不限制开发人员的创造力。

3.1.3 环境模型

愿景陈述与利益相关者的目标定义了系统或产品为实现其目的而应该满足的总体要求。另一方面，环境模型则体现了另外不同的视角，因为它们的目的是描述运行系统或产品的环境（上下文）的特定属性。

系统或产品的需求通常是在考虑环境假设的情况下特定的。环境模型是一种结构化的方法，用于记录有关环境的相关假设。明确这些假设有助于为整个团队和其他相关的利益相关者建立对系统或产品运行环境的共享和一致意见。

在不确定的情况下，如果环境模型中记录的假设被证明是正确的，即环境模型正确地反映了系统或产品实际操作环境，那么意味着定义的需求将是正确的。

环境模型是一个强大的制品，因为它们能清楚地区分要开发的系统或产品及其环境，例如，相邻系统和人类用户（参见 [CPREFL2022]）。通过使用这种差异化，功能可以被分派。

这样，就可以区分系统或产品本身的责任和上下文中相邻系统或人类用户的责任——并通过协作以实现总体愿景。因此，环境模型还可以用来澄清和指定要开发的系统或产品的外部接口。

环境模型可以使用不同的格式进行文档化，例如用于结构化系统分析的环境图、用例图、SysML 块定义图、UML 组件图或 UML 类图。只要能清楚地将待开发的系统或产品，以及环境中的与人员或系统的外部接口区分开来，使用任何标记方法都是合适的。“另外，标记方法还必须要以合适的详细程度来记录这些元素和待开发系统或产品之间的关系”，以便在适当的详细级别上进行开发。即使是简单的手绘框和线条图往往也是很实用的。

环境模型可以独立于文档的形式，它是一个非常有价值的制品，值得推荐在敏捷开发过程中使用。环境模型界定了区域（在范围的边界内），在这个区域内分析师可自由决策，而外部接口（即系统范围和环境之间的边界）必须与相邻的系统进行协商。

3.1.4 需求

需求必须根据愿景和目标来获取，并受到环境模型的限制。通常，需求被区分为三种类型：功能性需求、质量需求和约束。 [CPREFL2022].

3.1.5 需求的粒度

利益相关者通常在不同的粒度级别上交流他们的需求：“从总体业务目标这样的粗颗粒度需求到指定预期系统功能的细颗粒度需求”。因此，功能和质量需求（参见 3.1.8）可以（也应该！）在不同的抽象层次上进行讨论和记录。

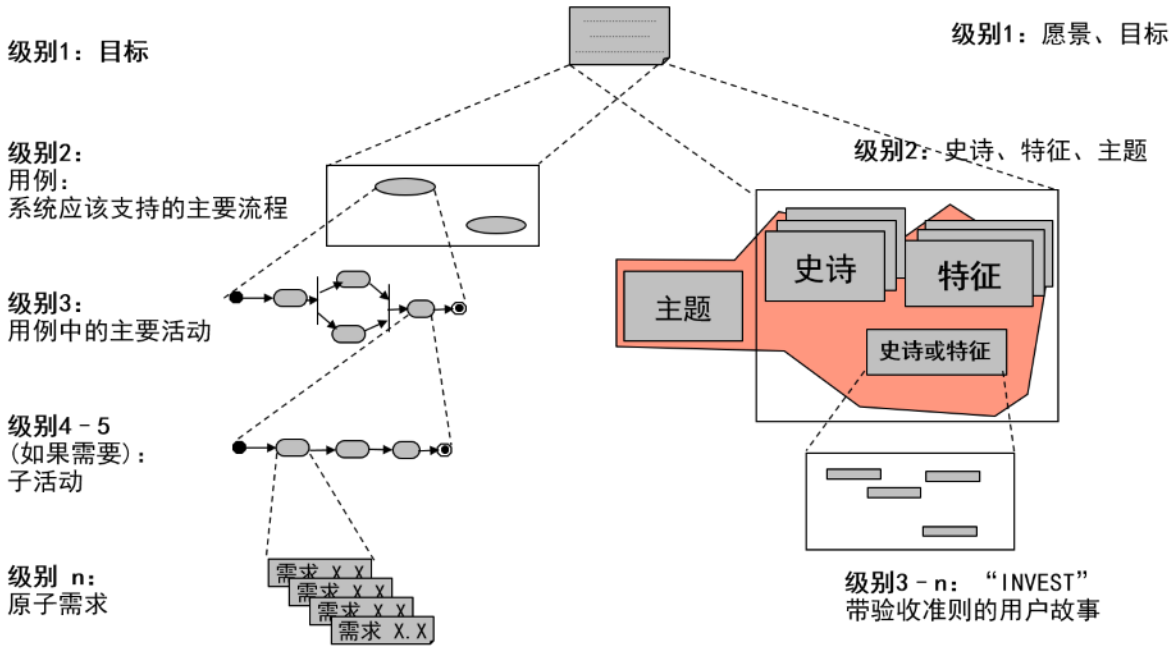

图 1：需求粒度：用例分解和敏捷变量、用例、用例规格说明和活动图

非敏捷方法论的一种典型方法如图图 1 左部表示。在这种方法中，用例首先用于细化愿景和目标，并提供预期系统功能的子模块（参见图 1—第 2 级）。图片仅展示了一个将需求进行高级别分组的例子。其他的例子包括按特征、业务对象、数据流或现有解决方案的组件进行分组。

在下一个粒度级别上，每个用例都被详细描述，以描述实现该功能所涉及的步骤。根据复杂程度，有几种备选方案可供选择：

1. 口头描述（纯文本）
2. 用例模板（半结构化格式或叙述流程）
3. UML 活动图（或任何其他类型的图形功能模型，如数据流图、BPMN、环境图、活动图、状态图、业务对象模型等。）

图 1 中第 3 级就是用 UML-活动图来描述用例的例子。

基于问题的复杂性，可以提供进一步的粒度级别。例如，将活动分解为子活动（参见 图 1 中的第 4 级）或单个活动的文本需求规格说明（参见 图 1 中的第 5 级）。

由此，粗粒度的客户需求被不断地细化为预期解决方案功能的细粒度规格说明。

注意，上述的方法说明的是一个自上而下的需求分析和分解的过程。这种有点理想的观点有时候是不现实的。实际上，可以首先在解决方案细节上表达需求，随后再建立更加笼统的目标。因此，现实情况下，需求分析的过程可以是自上而下，也可以是自下而上，或者是这两者的组合。

关键的一点是在不同的抽象层次上有不同的方法和技术可以支持这些讨论，在需求的总体知识增加的情况下，建立一个有意义的需求层次结构。

有一种选择可以用来描述低复杂度的用例，即只用几句话来阐明所要执行的必需步骤。对于中等复杂的用例，其他选项可能是一个不错的选择，比如用例模版。因为用例模板允许对更复杂的过程步骤进行描述，而这些步骤是特定用例的一部分。这种半结构化格式有助于理解从具有明确指导准则的用例中获取的需求，这样的用例中包含明确的书写指南。然而，如果有并行活动和多个场景的高度复杂性，那么用例模板有其局限性。

最后，需求工程师可以选择带活动图的图形格式，用它可以描述多个场景。在图中可以表示多个并行和连续的步骤流。这种用例描述的形式允许在一个图中描述复杂的用例流程。

一个总是引起讨论的问题是，需求是否被描述得足够详细。这里的主要关注点是团队内部的协作。当团队（尤其是实现需求的人，即开发人员）已经理解了需求时，则需求已经被足够详细地描述了。建议提前为此制定明确的标准。也请参见 就绪和已完成的定义。

敏捷术语：史诗、主题、特征和用户故事

在 图 1 的右边，使用敏捷术语，如史诗、主题、特征和用户故事，展示了一个等效的方法。

在这里，高级业务目标和复杂的功能被获取并为史诗或特征，并按主题分组。这样的主题可能非常复杂，例如，需要一个 Scrum 团队经过多个冲刺来开发 [Griff2015]。

然后将这些复杂的主题分解为更细粒度的用户故事。前面已经讨论过作为一个“好”用户故事的标准（2.2）。他们应该遵循就绪的定义[Kron2008]，比如满足 INVEST 标准[Wake2003]。特别是“E”、“S”和“T”是很重要的：它们应该是可估算的，小到可以在一个迭代里完成，并且是可测的。

再次需要说明的是，一个项目是否遵循严格的自上而下的过程来将史诗细化为特征，然后再细化为用户故事，或者是否是自下而上地先收集用户故事，随后形成共同的主题和史诗，这将取决于项目类型以及参与的利益相关者。无论是哪种方式，敏捷都提供了制品以对其全景和待开发的需求进行讨论和优先级排序。

3.1.6 图形模型和文本描述

无论粒度如何，功能性需求的标记方法始终有多种选择。功能性需求可以通过以半结构化的文本形式（比如基于模板的用户故事）来表达，也可以用一种简单的自然或形式化语言（如 Gherkin）来表达，清楚地说明解决方案应该做什么。

众所周知，自然语言有时不够精准和明确，因此已经开发了许多图形符号来克服这种歧义，或显示相互依赖关系，例如数据和过程之间的相互依赖关系。例如 UML 活动图、BPMN 图、流程图、状态图、顺序图等等。

大多数基于图形的标记方法强调不同的方面。活动图或 BPMN 图非常适用于显示连续步骤、替代或循环的线性过程。相反，每当异步事件会对流程产生影响时，使用状态图则是完美的选择。顺序图对于“举例说明”是最适配的，因为它们显示了交互的具体场景，而不需要真正地完成。

对过程和数据、数据与交互或过程和接口进行建模都是各有利弊的。它们是相互关联的（数据支持过程），所以它永远不可能是一个非此即彼的情况，只建议使用这个模型或那个模型， 这样的做法会适得其反。虽然文本需求更容易被许多利益相关者理解，但它们也很容易被不同的读者误解或误读。图形化模型提供更多的形式，从而避免了不同的解释或误解。风格的选择应该由需求工程的关键目标来决定，一方面确保所有利益相关者之间的共同理解，另一方面防止不完整性和误解[GoAk2003]。

3.1.7 术语、术语表和信息模型的定义

功能性需求如果没有对这些句子和图形模型中使用的所有术语的清晰理解，是不完整的。

因此，在任何需求制品使用的所有相关业务术语的集合是必要的制品，但是在仅关注业务需求时很容易被忽略。

这个制品的最小形式是一个（文本描述的）业务术语和缩写的列表，通常按字母顺序排序，以便更容易查找。这有时被称为字典、术语表或定义列表。

当业务非常复杂时，这个术语表可以通过将简单的术语分组到类（或实体）中，并以图形格式显示所有术语的概述。这些制品就会被称为数据模型、信息模型、实体关系图或 UML 类图。除了术语的定义之外，这些模型还包括这些实体之间的相关关联，即术语之间的静态关系。

正如上面所提到的，产品待办列表通常由史诗、特征和用户故事组成，强调所需的功能，而削弱了对业务术语的定义。然而，即使在敏捷方法中，对业务术语的清晰理解也是必要的，以便对诸如史诗、特征和用户故事之类的制品中使用的术语建立共同一致的理解。

3.1.8 质量需求和约束

除了功能性需求（指定系统或产品需要提供的功能）之外，质量需求和约束对于待开发系统和产品的成功也是至关重要的。传统上，质量需求和约束都包含在“非功能性需求”中[CNYM2000]。几十年来，需求工程专家一直强调这些“非功能”需求的重要性。尽管术语“非功能需求”在实践中仍然经常使用，但作为质量要求和约束的统称术语，IREB 使用了更具体、更精确的类别“质量需求”和“约束”。 [Glinz2022]。

质量需求与系统或产品需要展示的特定质量有关，例如，关于性能、可靠性、（功能）安全性、（信息）安全性或易用性等[ISO25010]。在以用户故事形式强调功能性客户需求的情况下，敏捷方法存在一个风险，即没有对质量需求进行明确阐明。质量需求无法被定义为可以在一个迭代里完成的用户故事：而描述的是正处于开发过程的产品新出现的属性，因此需要对所有用户故事进行持续测试。例如，需求工程质量方面的清单（参见[CPREFL2022]）应该是很有用的。另一种选择是将所有质量需求纳入完成的定义（DoD），确保满足所有质量要求，从而使待办项成为潜在可部署增量的一部分 [CPREALAGILE2022]。

众所周知，质量需求难以通过重构来构建到现有系统中，因此，在此过程中尽早考虑这些方面是非常重要的。

约束定义了针对待开发的系统或产品的解决方案空间的总体限制 [Glinz2022]。约束有多种形式：组织约束（预算限制、紧凑的时间表、规定的开发过程等）、技术约束（需要特定的数据库系统、使用特定的编程语言、选择的框架等）或系统将在其中运行的环境自身的约束（标准、规范、法规等）。

与质量需求类似，太晚意识到关键约束的代价可能非常昂贵，因为很多这些方面不能增量地添加。规划和设计决策依赖于对这些问题的良好理解，因此在此过程中，尽早识别关键约束是至关重要的。

与功能性需求类似，质量需求和约束可能存在于不同的抽象级别，并且应该记录在产品待办列表中，向团队公开，并在每个迭代中进行测试。因此，将约束添加到“完成的定义”中并通过自动化回归测试的形式来进行确认应该是有用的。

3.1.9 验收标准和适配标准

如果需求的实现无法被确认、检查或测试，那么任何需求类型的价值都是有限的。因此，每个需求都需要一套可以测试的标准，以检查需求是否已经实现。另外，这样的标准还可以通过描述为期望的具体术语来帮助理解（并鼓励反馈）。

所使用的标准类型与需求的粒度级别相匹配：

- 在较高的抽象层次上（愿景、目标和史诗），通常需要成功标准［SAFe］来进行定义，因为只有这样才有可能度量是否提供了某个所需要的功能或能力。
- 在较低的抽象层次上，可以用验收标准来描述如何根据需求对解决方案进行测试，以获得认可。

非敏捷和敏捷方法论一致认为需求必须是可验证的。非敏捷方法论经常使用诸如“质量标准”或“适配性标准”或普通的“测试用例”这样的术语；在敏捷中，术语“验收标准”（用于用户故事）或“成功标准”（用于史诗或主题）更为常见。

3.1.10 就绪和已完成的定义

验收和适配标准属于并且是业务需求、制品、“就绪定义”（DoR）和“完成的定义”（DoD）的一部分；同时，验收和适配标准支持正式的开发过程，并确保需求和产品增量的质量。完成的定义（DoD）是 Scrum 指南[Scrum2020]的正式组成部分，被描述为对增量（Increment）的承诺。完成的定义（DoD）在敏捷开发流程中起到质量保障的作用，而需求评审（DoR）则是一种最佳实践，有助于创建有价值的需求并避免团队因不合格的需求而负担过重。就绪的定义（DoR）帮助需求提升到正确的详略级别，并为产品负责人、需求工程师和开发人员之间进行的协商提供足够的信息。

3.1.11 原型 vs 增量

另一种处理需求的方法是原型，因为许多对系统或产品有需求的利益相关者并不想为了定义产品而进行编写或阅读文档，他们希望立即获得成功。对于这些人来说，了解他们的需求并获得反馈的最好方法是通过运行的系统来演示系统功能或能力。

一种方法是通过使用最小、增量的产品。需求工程提出了两种类型：“水平”产品增量，通过演示更多的变化/变体来进行需求确认（“做更多正确的事情”）；“垂直”产品增量，以验证开发方法是否正确（“做这一正确的事情”）。

通过提供与工作系统的直接交互，原型对于获得反馈非常有用。例如,反应时间等易用性属性是很难被确定下来，但却很容易在使用系统过程中被识别出来。然而，原型可能也会成为让用户感到沮丧的来源，因为他们相信开发已经完成了；对于开发人员来说，原型经常被后来更好的技术所替代，从而被抛弃。

敏捷方法试图通过立即开发“真实”系统或产品的高质量增量来规避一次性原型。敏捷致力于短迭代，交付可演示的产品增量，而这些增量反过来可以提供更多需求的信息。尽管重构的目的不是抛弃已开发的代码，但却是一种敏捷的良好实践，它是根据用户反馈改变功能或质量需求的一种解决手段。

敏捷中的术语“探针 spike”指的是一种开发迭代，它的明确目的是理解复杂的领域（例如，系统架构），从而降低风险。这个术语虽然没有精确定义，但可以指确认一个任务或整个迭代。原型设计是一种有效的确认技术，与其他敏捷迭代不同，它的主要目标是获得知识而不是生产可工作的代码。

3.1.12 关于制品的总结

正如在最后一段中提到的，一些制品对于成功的开发非常重要：

- 从愿景或目标开始是一种很好的实践。
- 自始至终都能识别和了解最重要的利益相关者是一种很好的实践。
- 明确地设定系统范围并从环境中界定范围是一个很好的实践。

尽管非敏捷的需求工程师和敏捷的需求工程师可能使用不同的术语，但他们一致认为，有必要了解功能和业务术语，以获取功能性需求（包括功能和数据）。

此外，敏捷和非敏捷方法论应该识别质量需求和约束，因为它们可能会深度影响设计决策。忽视他们或者太晚了解他们可能会导致大量返工以及不符合客户期望的产品。

良好的需求工程应确保没有任何相关问题被遗忘。因此，非敏捷方法论通常会坚持记录许多系统需求相关的潜在领域。

敏捷方法论使用相对非正式的制品并用直接的交流和快速的反馈来替代缺失的文档，这是通过增量产品开发或原型实现的。

敏捷宣言[AgileMan2001]中经常被引用的第二条原则正好揭示了我们刚才所讨论的内容：

„... 通过这项工作，我们建立了如下价值观：……工作的软件胜过详尽的文档……”

我们确信，如果这些组织认真考虑什么需要书面记录在文档中，什么需要讨论，什么可以以原型化形式或者以增量形式表达出来，任何组织都将从中受益。当文档、通信和原型/增量开发根据公司的约束和文化进行平衡时，将会取得最好的结果。第 4 章将更多地讨论如何用迭代活动平衡前期需求（和架构）活动。

3.2 需求工程@敏捷的技术（L2）

在 3.1 中，您已经了解了重要的需求制品。在这个知识单元中，您将了解在需求工程中要执行的关键活动。CPRE 基础级手册 [CPREFL2022] 以如下方式组织这些活动：

- 需求获取
- 需求记录
- 需求的确认与协商
- 需求管理

下面的小节从敏捷的角度讨论这些活动。

3.2.1 需求获取

敏捷开发中的需求工程是建立在利益相关者（包括开发团队）之间紧密的沟通基础上进行需求的获取。团队成员之间非正式的、直接的沟通本身可以认为是访谈和头脑风暴的一个很好的结合。像极限编程（XP）里的“现场客户”这样的技术可以在产品负责人和利益相关者（通常是细化待办列表中事项的开发团队）之间对话的方式也同样能获得成功。无论何种情况，其目的都是要更加深入地了解真正的需求。

比起敏捷开发环境中通常讨论到的技术，需求工程提供了更广泛的技术来发现和获取需求。这些技术包括问答技术（Q/A）（不仅包含访谈，还包含问卷调查），观察技术、基于制品的技术（需求的复用、系统考古等）[CPREFL2022]，以及创新型技术，比如头脑风暴和设计思维。这些技术支持以用户故事的形式表达的需求，这是结构化讨论的基础，而不是对于实现的建议。

正如在第 3.1.11 章中讲到的，原型和产品增量是深入了解需求的另一种方法。一旦产品增量在冲刺评审或演示会议中被展示出来，一些新的想法可能会产生并直接被纳入产品待办列表里，并由产品负责人进行优先级排序。

学习非敏捷需求工程中不同的需求获取技术以及如何慎重选择获取技术的组合，可以给敏捷需求工程带来很多益处。尽管利益相关者之间深入的沟通和对产品增量的迅速反馈是非常好的方法，但需要获取的需求还不仅于这些。举例说，如果有上百个或上千个利益相关者，仅有口头沟通是不够的。当试图获取一些创新需求，或者利益相关者甚至没有意识到这种可能性的需求时，就需要创造性技术。

当在紧迫的时间限制下工作时（即没有足够的时间进行深入沟通），snow cards 可能是最有效的技术。当开发新技术时，探针（给定时间盒内，探索潜在解决方案的简单增量）是很好的技术。

3.2.2 需求记录

敏捷模式下的需求被放在待办列表中。需求可以用标注了商业价值和优先级的故事卡的形式进行编写。这些可以组织在故事地图中，也可以分解成更简单的故事。卡片背后的原则是，卡片的大小限制了所写的内容，这样有助于将关注点放在核心细节上。

尽管如此，文档化需求仍然被认为是促进所有利益相关者之间交流的重要活动。如果详细信息是通过口头传达的，文档的形式主义可以被最小化。

定义文档的适当详细程度取决于很多因素，如项目或产品的规模、涉及的利益相关者的数量、法律约束或项目的安全关键性。基于这些因素，敏捷开发过程尽量避免文档记录太过详细，并找出文档记录中有用内容的最小集合。

虽然使用“活跃的”产品待办列表是处理文档的一种有效率方法，但它并不总是够用的。所以，我们来看一下其他类型的文档。

从需求工程角度，我们将文档划分为四种类型：

1. **满足合法性目的的文档：**某些领域或项目环境（例如，卫生保健部门的软件或航空电子设备软件）需要为特定受众提供特定信息（例如系统的需求和测试用例）的文档，以获得法律方面的批准。

编写满足合法性目的的文档的原则：合法的必要的文档必须从相应的法律或标准中派生出来，并且是产品不可分割的一部分。

2. **用于保存目的的文档：**系统的某些信息具有比最初开发成本更持久的价值。比如构建系统需要实现的目标、系统支持的核心用例、或在系统开发过程中做出的决策，例如排除某些功能的决策。用于保存目的的文档可以成为团队、产品或组织的共享存档。它可以减轻对单个团队成员的记忆能力的依赖，并且可以减少对以往的决策的再次讨论（例如：“为什么我们决定不实现它？”）.

原则：由团队决定哪些是用作保存目的文档。

3. **改善沟通目的的文档：**高效有效的沟通因其迭代性和较短的反馈周期而成为敏捷方法的重要工具。在实践中，有几种情形可能会阻碍直接的口头沟通：分布式团队、语言障碍或相关的时间限制。此外，信息有时是如此复杂，以致于直接的沟通可能会是低效的或具有误导性的。一个复杂算法的纸质原型或图表可以用来在以后再次使用，比如再次阅读。有时，与阅读源代码或评审软件相比，利益相关者更喜欢书面沟通。在这些情况下，文档有助于所有参与方之间的沟通过程并保存结果。

创建沟通目的的文档的原则：如果利益相关者或开发团队注意到现存文档中的一个价值，则编写新的文档作为附加的沟通手段。当沟通顺利结束时，文档应该被归档。

4. **帮助思考目的的文档：**在编写文档的过程中经常被遗忘的一个方面是书写过程是提高和支持作者思考过程的方法。即使在思考过程中文档可能会被遗弃，但对思考过程的改进和支持

作用还会一直存在。举例来说，编写一个测试用例就会强制作者去思考系统和操作者之间具体的交互，比如，异常和替换场景。因此，编写用例的过程可以被理解为测试自己对系统的理解和掌握知识的工具。

创建帮助思考目的文档的原则：思考者决定最能支持他/她思考过程的文档形式。思考者不需要证明为所选择文档形式的原因。当思考过程结束时，文档可能会被丢弃。

当这四种类型的文档在适当的环境下被有效的识别和应用时，敏捷方法的益处将会体现出来。总的来说，记录需求本身并不是目的，但是它应该促进利益相关者之间的沟通，尤其是在需求方/请求者（通常被产品负责人替换）和开发团队之间的沟通。

3.2.3 需求的确认与协商

尽管需求工程强调通过例如评审、走查、审查或基于视角的阅读等方法来确认需求，而敏捷方法则强调通过对有价值的产品增量的尽早的和频繁的反馈来确认需求。一个很好的例子是自动化回归测试，它提供了对开发和相关需求的持续确认。需求确认的目的包含识别缺失的、模糊的和不正确的需求以及有争议或有矛盾的需求，后两者可以应用协商和冲突解决技术来解决。

由于迭代式增量开发是敏捷方法中的一个关键策略，因此对文档的形式化确认需求会减少。它被利益相关者之间关于需求的持续协商所取代，因此冲突会被尽早发现和解决。

通过以集成产品增量显示快速结果的方法，也可以减少形式化的确认。如果增量不能满足所有利益相关者的所有需求，增量将以新需求的形式返回到产品待办列表中，并与所有其他待办列表项一起被评估和排序。

然而，产品待办列表的走查、业务价值的讨论、风险的讨论和需求的即时协商都是敏捷需求工程中有价值的技术。所有这些技术都可能在细化会议中使用，在这个会议中，产品负责人和开发团队（利益相关者如果有时间的话）一起工作，使用持续细化原则找到实现所需的细化级别。

在 Scrum 指南的第一个版本中，所谓的产品待办列表细化会议只是间接提到。在当前版本中[Scrum2020]，需求优化被明确地提到是一种需求获取、需求记录和需求确认的活动。优化会议作为其中的一个重要部分，这个优化过程可以尽早发现潜在的问题，然后减少冲刺计划会议所需的时间。

3.2.4 需求管理

在传统的需求工程中，需求管理会关注随着时间的推移来处理需求的所有活动。这包括版本管理、变更管理、配置管理、可追踪性，以及添加属性，如状态、估算、优先级、相互冲突需求之间的关联，以及参与捕获、检查、签署、实现或测试需求的人员。

如前所述（参见 3.1.1），在敏捷开发过程中维护需求的关键制品被收集在待办列表中。与传统的需求管理不同的是，待办列表是被设计来保留所有尚未实现的需求的最新和最好的版本。当满足这些需求的产品被交付后，这些待办项通常会被立即删除或存档。

在待办列表中进行的需求管理活动包括：

1. 需求优先级设置：确定需求的业务价值以决定何时去实现它们。业务价值越高，需求优先级越高，因为敏捷项目试图首先交付最高的业务价值。影响业务价值的因素及由此产生的优先级排序在 CPRE RE@Agile 专家级模块中涵盖。 [CPREALAGILE2022].
2. 需求评估：确定实现这些需求需要的工作量。过高的工作量估算对于产品负责人传达了一个明确的信息：如果要达到就绪的定义（DoR，参见 2）的状态，还需要做更多的工作。因此，他必须将各个项目划分为较小的单元，以便进行估算。

这并不是说需求管理的其他一些传统活动不会在敏捷中发生。但是该类活动会被记录在待办列表之外。

如何在一个给定的环境中，确定什么样的需求管理活动是合理的，这需要需求工程师在最小化费用、允许早期交付工作方案和组织的长期需求，例如法律合规性、操作文档或者给新的开发团队成员的交接文档等方面取得平衡。

结论

需求活动，如获取、记录、确认和协商，和需求管理活动仍然需要在敏捷开发过程中进行。在敏捷和非敏捷开发之间，首选的需求获取和记录技术可能会有所不同，但是通过相互学习可以发现最佳的解决方案，因为它结合双方的优势，减少了浪费或开销。这种工作方法体现了 Scrum 的五个价值观（承诺、勇气、专注、开放和尊重）和三大支柱（透明性、检查和适应） [Scrum2020]。

当把需求工程的技术融入敏捷的世界中，同时把敏捷的理念和原则引入需求工程的技术中去的时候，开放和尊重尤其有用。

在这一教学单元，你已经了解到，即使在敏捷需求工程中，制品要比在产品待办列表里的用户故事更多，并且需求工程的关键活动不应该被遗忘的——而是基于 1 中的阐述的敏捷原则，它们可使用不同的侧重点和方法来执行。

4 需求工程@敏捷的组织方面（L2）

课时： 1 ½ 小时

学习目标

EO 4.1.1 理解组织结构与需求工程@敏捷之间的相互作用
EO 4.2.1 了解敏捷需求工程过程中与利益相关者的互动
EO 4.2.2 了解沟通和协作如何改善结果
EO 4.2.3 了解敏捷中的管理角色
EO 4.3.1 了解规模化的动机
EO 4.3.2 了解组织团队的规模
EO 4.3.3 了解团队间沟通的方法
EO 4.3.4 了解规模化的示例框架
EO 4.3.5 了解需求工程中规模化的主要影响
EO 4.4.1 了解确定前期与持续需求工程水平的标准
EO 4.4.2 了解待办列表项的正确详细程度
EO 4.4.3 了解需求工程@敏捷中确认需求的价值
EO 4.4.4 了解产品待办列表的正确更新周期
EO 4.4.5 了解如何为开发周期找到正确的进度

4.1 组织对需求工程@敏捷的影响（L2）

敏捷开发过程起源于制造/加工业和基于经验的过程控制（参见[TaNo1986]）。敏捷原则易于理解，敏捷实践和框架（如 Scrum）易于在初创企业或小公司等环境中使用。它们很难在更大的组织中实施，这些组织像是有生命一样，抵御每个入侵者。但是，另一方面，当组织发现应用这些原则和实践的好处时，会把它们与自己的基因结合起来，就像康威（conway）定律[Conw1968]所描述的那样。这样人们对敏捷管理（参见[ACP/PMI]）和敏捷组织（参见[Denn2015], [Appel2011]）等主题会越来越感兴趣，导致了对敏捷的讨论往往超出了（软件）开发的范畴。

以客户为中心、自组织团队、赋能和授权个人以及持续改进的理念在广泛的商业领域中都引起了共鸣。Scrum@Scale 框架是核心 Scrum 框架的最小扩展，它将模块化结构保持在 Scrum 框架的核心，并允许根据您公司的独特需求进行定制来实现规模化 Scrum。

在软件开发中领域中，很多敏捷开发流程的实施在构思阶段就失败了，是因为组织的其他部分无法自我改变以支持敏捷团队。

为什么需要进行组织变革呢？

举两个例子来说明为什么企业的其他部分也必须改变以支持敏捷开发：

1. .需求方的（子）组织必须能够提供足够好的需求（充分但又不是太过详细-遵循就绪的定义），才能保持稳定的速度进行开发。这与经常变化的需求结合，则需要持续不断的需求流。
2. 人力资源部门需要了解雇佣哪些人才能正确地支持敏捷团队。Scrum Master 职位的招聘往往是糟糕的例子，编程技能以及编程的证书都会作为招聘时的要求，而这其实是错误理解了三个 Scrum 角色的原则。

在 4.2 中将讨论在组织层面集成一个敏捷组织单元，例如，一个在非敏捷环境下的 Scrum 团队。同样，如果在产品的开发中已经引入敏捷过程，但在很多情况下并不一定代表在整个企业的工作都遵循敏捷原则。

在 4.3 中将讨论在解决一个复杂问题时需要一个以上的敏捷团队，在这种情况下对组织的影响。在此，主要的关注点仍然在 IT 组织以及与业务之间对接上。并没有明确考虑企业向完全敏捷组织的转变，因为它超出了需求工程的讨论范围。

4.4 将关注组织的时间安排，特别是分析何时应执行需求工程活动的问题。

4.2 非敏捷环境下的敏捷开发（L1）

4.2.1 与 IT 组织外部利益相关者的互动

开发组织在企业内部的作用是向企业客户（包括组织内部和外部客户）提供解决方案和服务。

敏捷将客户置于产品开发的中心。这意味着客户在整个产品开发生命周期中都要参与进来，对提供的增量积极地给出反馈，并接受符合业务需要的新需求。

随着客户的持续参与，需求工程也成为一个持续的过程（参见 2.4）。负责人，例如 Scrum 中的产品负责人，应该与客户进行公开和直接的交流，把倾听到的新需求和变更抓取到待办列表中。

在实践中，这种交流可以采取多种形式。如果客户每天都能够与团队合作，进行最直接的沟通，且大多数是非正式的，那么只需要记录结果或决定即可。

重要的是要意识到存在开发人员无法控制的因素（例如，地理分离或者就是单纯地缺乏可用性），在这种情况下，直接互动并不总是可行的。一个更有效的沟通形式应该是邀请客户代表定期参加计划和评审会议，或在早期阶段，通过定时、频繁的会议（如设计思维 design thinking 或设计冲刺 design sprint，参见 1.5）把结果输入到待办列表中。

4.2.2 产品组织 vs. 项目组织

敏捷促进了组织内的直接的、开放的和无层级的关系，并在产品开发过程中允许更大的灵活性。在基于自上而下的管理结构的传统大企业中，高度重视计划和可预见性， 如项目和资源的计划在一个企业中是必须要有的。

传统的软件开发是基于项目的，意味着它是作为一系列临时性任务进行的，以产生明确的产品、服务或结果为目的（参见 [PMI]）。具有共同目的或目标的相关项目组通常称为程序，而组织内项目和程序的整体规划和控制称为项目组合管理。

其实，在产品开发中，敏捷方法更多的是以产品为中心。维护产品改进的待办列表，并在持续改进的迭代过程中去实现。敏捷并没有定义结束日期。原则上，只要改进能带来价值，效益大于投入/成本，就应继续工作。

虽然这些方法并不是互斥的（项目的系统范围可能是交付特定的产品，也可能是为改进而建立的项目），但不同的视角和术语可能是敏捷软件开发组织和非敏捷组织之间误解的根源。

解决这种误解的一种方法是，虽然软件开发本身以严格的敏捷方法进行，但项目组合管理的功能提供了更高层次的规划和控制，可以结合起来使用（也参见 4.3）。关键是在计划实现的业务目标和项目组合层面的特征之间建立概念的桥梁，迭代和灵活地交付各个软件特征。

需求工程提供了必要的概念和方法来区分不同抽象级别上的需求，组合和程序级别上的业务需求以及适合于待办列表的衍生的和详细的软件需求。需求方法是把更详细和更精确的需求转变为精确的开发顺序，将需求作为讨论和调整的共同基础，并为当前的计划提供详细信息。

4.2.3 敏捷环境下的管理角色

学科和团队：IT 部门的员工传统上是按学科组织的：开发人员、测试人员、需求工程师、业务分析师、项目经理等。项目团队是由这些拥有不同技能的人员组合而成， 为固定时间的承诺而服务 。敏捷，特别是 Scrum，促进了跨功能团队的想法。除了产品负责人和敏捷教练的专家角色之外，所有团队成员都应该有能力扮演不同的角色，并根据需要支持需求工程或测试活动。同样由于这个原因，这些团队成员在 Scrum 中通常被称为“开发人员”。团队的目标是能够充分满足客户的需求，不管涉及的技术或组织因素如何。这点可以通过在开发团队中（首选解决方案）或在开发团队之外（在现实中通常作为产品负责人 Product Owner 的支持）拥有需求工程的能力来实现，而后者已不是 Scrum 框架的一部分。

IT 经理：IT 经理（在 [SAFe] 中称为“开发人员-people developers”）的职责是在他的组织中找到专家和通才之间的平衡，并帮助团队按所需能力的要求组织团队。康威（conway）定律 [Conw1968] 指出，团队结构将是产品结构（组件）或 IT 系统结构的镜像。这在规模化团队数量时更为重要。如果一家公司按照组件或系统组织团队，通过增加团队数量来进行规模化并没有用，因为它创建了更多的依赖关系。规模可以通过扩展团队成员的数量来实现，但只能在一定程度上实现，因为沟通成本也将急剧增加。

跨功能团队承担了从前端到后端交付增量的所有能力，可以轻松地进行扩展 — 但构建起来并不容易，而且也很耗时。

产品负责人与项目经理：正如前面所讨论的，产品负责人通过给产品开发团队排列业务需求的优先级，扩展了需求工程师的职责。产品负责人必须被授权（了解）并有权做出商业决策。在敏捷方法中，由于开发团队在组织范围内高度自组织，并且只需要详细到足以支持迭代开发的需求，因此项目经理先前做出的部分决策变得不再必要。开发团队可以自行组织工作，包括将任务分解并分配给团队成员。

敏捷的 IT 经理需要的是设立清晰的敏捷开发愿景和使敏捷开发文化畅通交流，这样才能成功。

满足业务期望的同时取得正确的平衡绝非易事。下面的中讨论了一些成功的标准 4.4。

需求工程（RE）的相关性：作为一个例子，前面的模式也适用于 Scrum。根据需求工程原则工作的人员可以是团队的一部分，因此不会单独命名。或者，他们可以自己组成一个团队，作为一个团队支持一个或多个产品负责人。这两种方法都有各自的优点，可以在不违反敏捷原则的情况下结合使用。需求工程将成为敏捷开发成功的支柱。

4.3 通过规模化处理复杂问题（L1）

4.3.1 规模化的动机

直接、日常、无层级交流的敏捷价值观通常反映在小型的、紧密互动成长团队中，比如 Scrum 团队，推荐由不超过 10 名 Scrum 团队成员（开发人员+产品负责人+敏捷教练）组成。团队成员最好在同一个物理位置，并且在业务和技术方面实现跨功能。在较大的组织中，这种理想的敏捷开发的价值观可能是不可行的，原因有很多：

- 复杂的问题可能涉及来自不同业务领域的利益相关者和知识，而这些很难纳入到单一团队。
- 复杂的问题可能涉及一系列技术专家和知识，而这些很难纳入到单一团队。
- 按照规定的发布日期划定的项目系统范围超出了单个团队所能达到的速度。
- 全球型企业的员工可能分布在各个不同地域。

规模化敏捷是用来描述多个团队（主要是 Scrum 团队）在一个产品/解决方案上协同工作、共享共同目标的情况。规模化敏捷方法需要确定如何组织 Scrum 团队，以及如何协调团队之间的沟通。目标是能找到实现处理复杂问题的有效方法，同时尽可能多的保留敏捷优势。

4.3.2 组织团队的方法

有一个不得不面对的问题是，组织应该如何安排一个团队的规模，使团队能够跨功能（最小规模），但同时在沟通和协调方面也是最有效的（最大规模）。按照功能来组织团队（例如，专门从事指定业务领域的团队，甚至是业务区域内的一个特定特征）具有将业务知识集中在单个团队中的优势。这可以减少需求的获取，例如，减少与团队直接相关的业务利益相关者的数量。

这种方法的一个缺点是，提供业务领域的端到端的功能可能涉及几个不同的技术专长，如用户界面设计、流程引擎、数据库和核心平台（如 ERP 或大型机），这可能会过度冲撞有效团队最大规模的界限。

组织开发的另一种方法是遵循技术路线，组织专门从事技术组件或平台的团队。组件或平台团队的优势在于他们会对各自的技术领域有深入的了解。缺点是这样的方法依赖于团队之间的协作来交付整个产品增量。

规模化框架（如那些在 4.3.4 中命名的框架）展示了处理这种情况的方法。需求工程活动必须与框架保持一致，才能实现对可交付产品的共同理解。

在这种场景下，需求工程在将业务目标和需求分解成子系统需求方面发挥着特别重要的作用，然后将其分配给各个团队，其次是通过开发集成的解决方案，使业务价值得以实现。

所有团队类型的混合类型可以提供最佳和最实用的解决方案，受益于功能团队的想法，但同时也要考虑到公司特定的约束（具体请参见 [CPREALAGILE2022]）。

4.3.3 组织沟通的方法

通过回答问题，“几个团队可以一起高效工作？”，我们可以区别出两个不同的方法：

1. 采用来自单一团队提供的方法
2. 引入其他的概念去组织沟通和责任

采用来自单一团队提供的方法：第一种方法遵循以下想法，不需要额外的制品和角色；并且应该使用单一团队的现有技术支持多个团队之间的沟通。其他角色和制品将与敏捷思维相悖，并导致组织更复杂。根据“保持简单”的基本原则，不应创建基于新角色的开销。团队之间的沟通和协调通常由团队的产品负责人发起，但由团队代表完成。构建类似实践社区这样的形式，使团队中的团队成员能够分享经验并协调整个流程。

引入其他的概念：第二种方法建议将较大的问题分解为较小的问题，通过不同的角色来负责不同的抽象级别。因此，应该为不同的抽象级别（例如，业务史诗、架构史诗、投资主题、特征、用户故事）引入额外的制品。

根据所使用的框架，还引入了其他角色，负责不同的抽象级别（例如，项目组合经理、产品经理和首席产品负责人）。由于复杂性的增加，还需要额外的制品和角色来管理不同团队之间的计划和沟

通，并实现每次迭代的集成结果（例如路线图和发布经理）。此外，还必须组织特定的会议，以促进新的和已经确定的角色之间的沟通。需求工程提供了许多技术，可以帮助细分大问题并支持不同抽象级别的新角色（例如环境建模、目标建模）。

以上两种方法都有各自的优缺点，每个用户或企业都需要根据自己的业务案例找到最佳方法。

4.3.4 需求工程@敏捷规模化的示例框架

支持 Scrum 和敏捷规模化的框架：有许多不同的框架可用于支持这些方法，并且由于规模化敏捷的重要性日益增加，这个数字也在快速增长。您将找到以下最常见的框架：

- **规模化敏捷框架（SAFe）**
  SAFe 是在企业级规模上实施精益敏捷软件和系统开发的成熟模式的知识库。 [SAFe]
- **大规模 Scrum（Large-Scale Scrum/LeSS）**
  LeSS 是一种 Scrum，它应用于多个团队与一个产品负责人在同一产品中的协同工作。 [Less]
- **Nexus**
  Nexus 是一个针对规模化核心的框架：解决跨团队的依赖和集成的问题。 [Nexus2015]
- **Scrum of Scrums**
  Scrum of Scrum 是一种使用 Scrum 来协调多个团队的技术。[SofS] 每个团队指派一个人（大使）代表他们参加协调会议，协调会议通常每周举行两到三次 [CPREALAGILE2022].
- **规模化 Scrum（Scrum@Scale）**
  Scrum@Scale 框架是核心 Scrum 框架的最小扩展，它将模块化结构保持在 Scrum 框架的核心，并允许根据您公司的独特需求进行定制来实现规模化 Scrum。[SatS].

4.3.5 规模化对需求工程@敏捷的影响

以上讨论的抽象层意味着需求工程活动由更多的角色来管理，例如首席产品负责人、产品经理、项目组合经理、业务分析员。每个角色将为他们各自关注的领域（史诗、特征、用户故事和它们之间的可追踪性）创建相应的需求工程制品。因此需要额外的会议来进行需求工程活动（例如，故事时间、特征时间、系统演示），而与利益相关者的沟通是由组织内不同级别的不同角色执行的。

由于 Scrum 本身不容易按其自身的范围进行规模化，因此在确定多个敏捷团队中如何适当地分解和分配总体需求，以保持规模化方法可用时，需求工程起着关键的作用。需求工程技术有助于构造问题分析，并将粗粒度需求细化为细粒度的需求，如适合于单个团队的特征和用户故事。应用适当的抽象和不同分析视角的结构化方法，将为半自主敏捷开发团队中的任务细分提供良好的基础。

以上的内容尤其适用于在需要尽早发现和讨论需求内的依赖关系，以避免开发。

如果团队或其他角色在不同的地方工作，将会带来更多的挑战。管理沟通的复杂性不仅仅因为潜在的时差或不同的语言而增加。在大多数情况下，主要挑战是基于不同的文化。由于沟通是产品负责人和产品经理的主要任务之一，这对需求工程相关角色所需技能有重大影响。

4.4 在规模化环境中平衡前期和持续的需求工程（L1）

在一个非常抽象的层面上，敏捷方法论可以被描述为一个连续的、迭代的过程，在这个过程中，系统基于待办列表项进行增量开发。从需求工程的角度来看，可以识别出五个能驱动这个过程的参数（见下面的章节）：

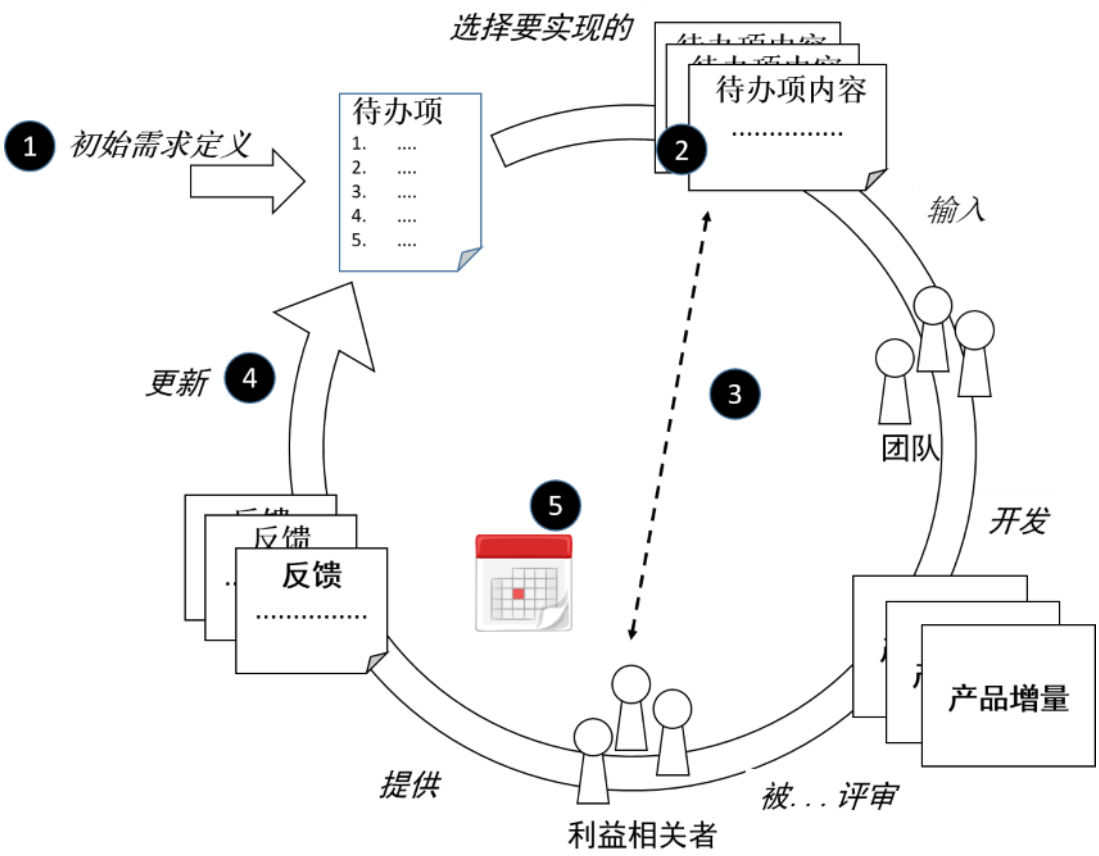

图 2：持续需求工程过程

4.4.1 初始需求的定义

在开始连续开发过程之前，必须先创建初始待办列表（参见 [TaNo1986]）。这种待办列表的初始定义通常被称为“预定义”。需求工程有时被理解为规定这种预定义是对所有需求的完整和详细的规格说明。对于某些流程模型或领域来说，情况可能是这样的，但并非总是如此，而且在敏捷开发中并不常见。

初始定义的待办列表基线规定了开始连续开发过程的第一次迭代所需的信息汇总。何时开始迭代过程是一个关键的决定。系统和环境可能会导致风险，例如，需求的不确定性（一开始没有足够的信息）可能导致额外的延迟、成本或其他可能导致项目失败的潜在风险等。同样，起步太晚可能会对上市的时间和适合最终用户要求的特定时间点产生负面影响。

需求工程告诉我们，以下这些需求应该尽早地获取和弄清细节：被识别出对结构有很大影响的需求；或者对解决方案的整体可行性有很大影响的需求；或对基础设施和硬件的关键选择有很大影响的需求。这类结构相关问题实际上应该在开发的第一次迭代之前获取并分析。这样在连续的迭代过程中，需求得以精炼并降低其变化带来的影响。

在敏捷的迭代和增量过程中，需求工程变成了一个连续的过程，它及时地传递和细化需求，并提供足够的细节反馈给开发周期。这个过程类似于一个用于粉碎石头的漏斗，大石头进入漏斗，并在加工完成时粉碎成中等大小，最后变成小石块。

4.4.2 待办列表项的详细程度

待办列表项的详细程度限制了开发团队实现待办列表项的自由度（参见 [Pich2010]）。待办列表项的所有未定义的方面都留给团队来做决策，这能使得团队更有创造性，但必须保证实现还是在业务定义的范围内。

开发团队的能力可以作为一个经验法则来定义待办列表项的详细程度。如果开发人员有足够的能力和经验来决定待办列表项目的细节（例如，开发人员中有一位认证和授权专家），那么细节的决定权应该留给开发人员。参见 3.1.5。

4.4.3 待办列表项的有效性

在实施之前了解待办列表项的有效性有助于最大程度避免不必要的开发工作（参见[Denn2015]）。因为在已实现的软件中去识别错误或不完整的需求是一种非常昂贵的需求确认方法。

确定待办列表项的有效性与特定待办列表项的详细程度密切相关。只有当待办列表项足够详细时，待办列表项的有效性才能确定。因此，在实施之前花费在详细阐释和确认需求上的工作量，必须与实现该需求以及软件交付后验证该需求的预估工作量进行比较。

当通过可接受的工作量就能确定待办列表项利益相关者的需求偏好时，应该在开发待办列表项之前对这样的需求进行确认。

这种类型的需求（包括示例性确认方法）典型示例有：用户界面的总体设计（例如采用 UI 模型）、授权和认证机制（例如使用用例评审）、必须存储在系统中的数据结构（例如采用数据模型评审），以及针对现有系统接口的需求（例如活动图评审）。

高风险（例如关键业务功能、安全性关键功能、创新功能）或具有较高测试成本的待办列表项（例如，软件必须在昂贵的原型中测试），应在实施前进行确认。

4.4.4 待办列表项的反馈和更新

待办列表项往往基于审查活动的反馈进行更新，例如根据冲刺评审的结果进行更新。这种方法对于小规模或详细的需求是可能的，因为在一个或两个小型开发团队的环境中，可以快速地分析和把握变化产生的影响。但是在短时间内修改具有更大复杂性或具有多个依赖项的需求是不可取的。在这种情况下，修改待办列表项需要更多的时间，利益相关者在某些情况下可能无法参与，因此需要额外的分析。

另一个可能对待办列表项的修改产生影响的因素是组织的决策过程。在重大决策耗时较长的组织中（例如责任理事会每三个月只召开一次），持续细化的原则需要采取召集所有受影响的利益相关者进行会议的形式。这样的会议需要需求工程提供准备，并提供决策支持。

4.4.5 开发周期的时间安排

开发过程的最终参数是迭代的时间表或迭代周期。这对处理当前未开发的待办列表项时需要执行的需求工程活动有显著的影响。此外，迭代周期决定了将迭代结果传递给业务利益相关者或供用户进行检查的频率。

因此，更短的迭代周期增加了业务利益相关者的工作量，原因有三个（如，[Rein1997]）：

1. 业务利益相关者必须在整个迭代过程中保证对待办列表项工作的充分投入，以便为创建下一次迭代提供意见。
2. 业务利益相关者必须评审团队创建的结果，并提供反馈。
3. 业务利益相关者需要习惯在每天的业务和项目工作中，切换不同的背景。

较长的迭代周期减小了业务利益相关者的压力，但也降低了其对产品开发待办列表项的影响力。

迭代周期的定义必须考虑到业务利益相关者可投入的程度。因为业务利益相关者在系统开发中有其他职责，所以他们通常不是 100%投入到开发活动中。

根据经验，更短的迭代周期提供更频繁的反馈和更多的机会提前发现错误，因此较短的迭代往往加速开发活动。但是如果较短的周期会造成利益相关者的超负荷工作，那么应该找到折衷的迭代周期的长度，尽管这可能会对项目优先级产生相应的下调。

影响周期时间的另一个因素是待办列表项的平均大小和复杂性。更大或更复杂的待办列表项需要更多的时间来理解和分析。因此可以考虑增加周期长度，这样能够在一次迭代中处理更大或更复杂的待办列表项。例如，如果正在开发的系统处于早期阶段，则建议采用更长的周期给团队更多的时间来获得对系统的初步理解。当然周期时间必须平衡使用较短的周期时间以获得频繁反馈的目标。在决定是否采用更长或更短的周期时间时，团队必须在分析活动的需要与更早得到反馈的目标之间做出权衡。改变周期时间总是基于“审查和适应”的原则，记住过去的经验并不总是对未来的保证。改变周期时间应该在冲刺之前进行，而不是在周期内进行。此外，周期时间应尽可能保持一致，以降低复杂性并建立可持续的预测能力。频繁更改冲刺时长可能会导致项目不稳定，降低团队参与度，并极大地阻碍可行产品增量的交付速度。

5 术语及术语表定义（L2）

术语表定义了与需求工程@敏捷模块相关的术语。术语表可以从 IREB 主页 https://www.ireb.org/en/downloads/#re-agile-glossary 下载。

6 参考文献

[ACP/PMI] Agile Certified Practitioner: http://www.pmi.org/certification/Agile-management-acp.aspx. 最新访问 2024 年 6 月

[AgileMan2001] Agile Manifesto: http://Agilemanifesto.org, 2001. 最新访问 2024 年 6 月

[AmLi2012] Ambler S.; Lines, M.: Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise. IBM Press, 2012

[Ande2012] Anderson, D.J.: Lessons in Agile Management: On the Road to Kanban. Blue Hole Press, 2012

[Appel2011] Appelo J.: Management 3.0. Addison-Wesley Professional, 2011

[Beck2003] Beck, K.: Test-Driven Development by Example. Addison Wesley - Vaseem, 2003

[Beck2004] Beck, K.: Extreme Programming Explained: Embrace Change. Addison-Wesley Professional, 2004

[CNYM2000] Chung, L.; Nixon, B. A.; Yu, E.; Mylopoulos, J. : Non-Functional Requirements in Software Engineering. Springer Science & Business Media, 2000

[Cock1998] Cockburn, A.: Surviving Object-Oriented Projects. Addison-Wesley, 1998

[Cohn2004] Cohn, M.: User Stories Applied: For Agile Software Development. Addison Wesley Professional, 2004

[Conw1968] Conway, M.: How Do Committees Invent? Datamation 14(4):28－31, 1968. Article available at http://www.melconway.com/Home/Conways_Law.html. 最新访问 2024 年 6 月

[CPREFL2022] IREB e.V.: Syllabus CPRE Foundation Level, version 3.1. https://www.ireb.org/downloads/#cpre-foundation-level-syllabus-3-0. 最新访问 2024 年 6 月

[CPREALAGILE2022] IREB e.V.: Syllabus CPRE RE@Agile Practitioner|Specialist, version 2.2.0. https://www.ireb.org/downloads/#cpre-advanced-level-re-agile-syllabus. 最新访问 2024 年 6 月

[Denn2015] Denning, S.: How To Make The Whole Organization Agile. http://www.forbes.com/sites/stevedenning/2015/07/22/how-to-make-the-whole-organization-agile/#658d3f65135b, 2015. 最新访问 2024 年 6 月

[Dsch2015] d.school: An Introduction to Design Thinking - Process Guide. https://web.stanford.edu/~mshanks/MichaelShanks/files/509554.pdf, 2015. 最新访问 2024 年 6 月

[Glinz2022] Glinz, M.: A Glossary of Requirements Engineering Terminology, https://www.ireb.org/downloads/#cpre-glossary. 最新访问 2024 年 6 月

[Griff2015] Griffiths, M.: PMI-ACP Exam Prep. Rmc Publications, 2015

[GoAk2003] Gordijn, J.; Akkermans, J.M.: Value-based Requirements Engineering: exploring innovative e-commerce ideas. Springer, 2003

[High2009] Highsmith, J.: Agile Project Management: Creating Innovative Products. Addison-Wesley Professional, 2009

[ISO25010] ISO/IEC Systems and software engineering - Systems and software Quality Requirements and Evaluation. ISO/IEC Standard 25010:2011

[KnZK2016] Knapp, J.; Zeratsky, J; Kowitz, B.: Sprint: How to Solve Big Problems and Test New Ideas in Just Five Days. Simon & Schuster, 2016

[Kron2008] Kronfaelt, R.: Ready-ready: the Definition of Ready for User Stories going into Sprint planning. http://scrumftw.blogspot.de/2008/10/ready-ready-definition-of-ready-for.html, 2008. 最新访问 2024 年 6 月

[Less] Large Scale Scrum: http://less.works. https://less.works/resources/LeSS-brochure.pdf 最新访问 2024 年 6 月

[LiOg2011] Liedtka, J.; Ogilvie, T.: Designing for Growth: A Design Thinking Tool Kit For Managers. Columbia Business School Publishing, 2011

[Martin1991] Martin, J.: Rapid Application Development. Macmillan Coll Div, 1991

[Meyer2014] Meyer, B.: Agile! - the good, the hype, and the ugly. Springer, 2014

[MeMi2015] Mesaglio, M., Mingay, S.: Bimodal IT: How to Be Digitally Agile Without Making a Mess, Gartner 2015, https://www.gartner.com/en/documents/2798217. 最新访问 2024 年 6 月

[Nexus2015] Nexus Guide https://www.scrum.org/Portals/0/NexusGuide%20v1.1.pdf, 2015. 最新访问 2024 年 6 月

[Patt2014] Patton, J.: User Story Mapping. O'Reilly, 2014

[Pich2010] Pichler, R: Make the product backlog deep. http://www.romanpichler.com/blog/make-the-product-backlog-deep/, 2010. 最新访问 2024 年 6 月

[PMI] PMI Project Management Institute. http://www.pmi.org/. 最新访问 2024 年 6 月

[Popp2003] Poppendieck, M.: Lean Software Development: An Agile Toolkit. Addison-Wesley Professional, 2003

[Rein1997] Reinerstsen, D. G.: Managing the Design Factory - A Product Developer's Toolkit. Simon & Schuster, 1997

[Ries2011] Ries, E.: The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses, Crown Publishing Group, 2011

[SAFe] SAFe - Scaled Agile Framework. http://www.scaledAgileframework.com. 最新访问 2024 年 6 月

[SatS] Scrum at Scale Framework: https://www.scruminc.com/scrum-incs-scrum-at-scale-framework/. 最新访问 2024 年 6 月

[Scrum2020] Schwaber, K. & Sutherland, J.: The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game, July 2020. https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf. 最新访问 2024 年 6 月

[ShYo2006] Sheppard, J. M.; Young W. B.: Agility literature review: Classifications, training and testing. Journal of Sports Sciences 24(9): 919-932, 2006

[SofS] Scrum of Scrums https://scrumguide.de/scrum-of-scrums. Last visited June 2024, available in German only.

[TaNo1986] Takeuchi, H.; Nonaka, I.: The new new product development game. Harvard Business Review 64(1), January/February 1986, p.137-146

[Wake2003] Wake, B.: Invest in Good Stories and Smart Tasks, http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/. 最新访问 2024 年 6 月