Certified Professional for Requirements Engineering

基础级别

教学大纲

Stan Bühne
Martin Glinz
Hans van Loenhoud
Stefan Staal

3.2.0 | 2024 年 2 月 26 日

使用条款

1. 培训机构及个人可以用这份教学大纲作为研究讨论的依据，但必须注明本文的出处及版权。任何在广告中使用此教学大纲的人都需要得到 IREB e.V 的书面同意。
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，必须注明出处和作者以及 IREB 的版权所有。

© IREB e.V.

本文版权归 IREB 所有。未经作者或者 IREB e.V.事先书面授权，本文的任何部分不允许被复制修改，或者保存在可再读取的系统中，或以任何媒体形式（电子、机械、影印、录音等）传播。

鸣谢

本教学大纲最初是在 2007 年由 Karol Frühauf, Emmerich Fuchs, Martin Glinz, Rainer Grau, Colin Hood, Frank Houdek, Peter Hruschka, Barbara Paech, Klaus Pohl 和 Chris Rupp 创建。

他们得到了 Ian Alexander, Joseph Bruder, Samuel Fricker, Günter Halmans, Peter Jaeschke, Sven Krause, Steffen Lentz, Urte Pautz, Suzanne Robertson, Dirk Schüpferling, Johannes Staub, Thorsten Weyer 和 Joy Beatty 的支持。

版本 3 是由 Stan Bühne, Martin Glinz , Hans van Loenhoud 和 Stefan Staal 创建的主要修订版本。他们得到了 Karol Frühauf, Rainer Grau, Kim Lauenroth, Chris Rupp 和 Camille Salinesi 的支持。

在这次修订中，Xavier Franch, Karol Frühauf, Rainer Grau, Frank Houdek 和 Thorsten Weyer 提供了反馈， Wim Decoutere 和 Hans-Jörg Steffe 提供了补充反馈。

该修订由 Christoph Ebert, Barbara Paech 和 Chris Rupp 进行评审。

经 Xavier Franch 和 Frank Houdek 推荐，IREB 委员会于 2020 年七月 22 日批准了该大纲英文版本的发布。本中文版教学大纲翻译参与，按姓名首字母排：陈智迪、任亮、王翊宸。本中文版教学大纲评审参与者：刘海英、李紫源、夏勇。

感谢所有对本文做出贡献的人。

版权© 2007-2022 由上述作者共同拥有。并且授予 IREB e.V.国际需求工程委员会 (IREB)，卡尔斯鲁厄， 德国。

前言

2017 年夏天，我们进行了一项对现有需求工程认证专业人员（CPRE）基础级认证（v2.2）的相关性的调查。调查的目的是为了从培训机构以及作为需求工程师的 CPRE 认证从业者的角度获得关于该认证的实际市场相关性的反馈 [MFeA2019]。调查显示，现有的 CPRE 基础级教学大纲 v2.2 总体上仍然符合市场上最重要的需求，并向考生传授了相关的需求工程知识。然而，我们收到的反馈是有几种技术在实践中已不再使用，而其他更多适合迭代和适应性开发的技术却缺失。此反馈符合 IREB 自己对需求工程（RE）领域变化的看法。因此，我们对 CPRE 基础级教学大纲进行了重大修订，删除过时的内容并添加新的元素。版本 3 的教学大纲反映了当代需求工程的状态，涵盖了计划驱动和敏捷方法来说明和管理需求。

根据本教学大纲，申请 CPRE 认证的考生应具备一些具有计划驱动和敏捷方法的系统开发基础知识。

本文的用途

本教学大纲定义了由国际需求工程委员会（IREB）建立的需求工程认证专业人员的基础级别。本教学大纲为培训机构提供了他们组织教学材料的基础。学生可以使用本教学大纲为他们参加考试作准备。

教学大纲内容

基础级别是根据所有涉及需求工程专业人员的需要而编写的。包括的人员角色，例如需求工程师、业务分析师、系统分析师、产品所有者，或产品经理、开发人员、项目或 IT 经理，或领域专家等。

本教学大纲和相关手册使用"RE"来简称"需求工程"。

内容范围

CPRE 基础级传达了对任何类型的系统（例如：移动应用、信息系统或网络物理系统）都同样有效的基本原则。此外，CPRE 基础级不假定任何特定的开发过程，也不针对特定的应用程序域。培训

机构可以提供侧重于特定类型的系统、过程或应用领域的培训，只要完全涵盖此教学大纲的教学目标。

细化程度

本教学大纲的细化程度适用于国际统一的教学和考试。为了达到这个目标，

教学大纲包含以下内容：

- 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 / 认知等级

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。

较高的级别包含了较低的级别。请注意，术语表中指定为基础术语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 的主页进行下载：https://www.ireb.org/downloads/#cpre-glossary

教学大纲的结构

教学大纲由七个主要章节组成。每章涵盖一个教学单元（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。课时表示培训课程应为该章节投入的时间。培训机构可以自由分配更多的时间，但应保持教学单元之间的比例。章节中使用的重要术语列于本章的开头。

示例

EU4　　需求疏理的实践（L3）

课时：　　4小时30分钟

术语：需求来源、系统边界、系统上下文、需求获取、需求确认、利益相关者、卡诺模型、冲突、冲突解决

这个例子表明，第 4 章包含 L3 级的教学目标，需要 4.5 小时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（EO）会列在正文之前。编号显示它们属于哪一节。例如，教学目标 EO 4.2.1 出现在第 4 章第 2 节中。

教学大纲中主题的顺序

本教学大纲中的章节顺序构成了主题的逻辑顺序。然而，不是必须按此顺序教授这些主题。培训机构可以自由地按他们认为适合其教学理念的任何顺序教授材料（包括不同教学单元的主题交错）。

考试

本教学大纲是 CPRE 基础级资格认证考试的基础。

> 考试的一个考题可以包括本教学大纲的多个章节的内容。教学大纲的所有章节（EU1 至 EU7）都可能会考到。

考试的形式是不定项选择题。

考试可以在培训课程结束后立即举行，也可以与培训时间分开（例如在考试中心进行）。IREB 许可的认证机构列表可在网站上找到：https://www.ireb.org

版本历史

版本	日期	备注
2.2.0	2015 年 3 月 1 日	初始版本(基于英语版本 V2.2)
3.0.0	2021 年 7 月 1 日	第二版主要更新，以反映当代需求工程的状态，包括计划驱动和敏捷方法来说明和管理需求。
3.1.0	2022 年 11 月 1 日	修复了整个文档中的错别字和引用，以提高可读性。 EU 1 : ▪ 将学习目标 E0 1.3.2 移至 EO 1.1.2。 ▪ 更新了 E0 1.2.2 和 1.3.1 EU 3 更新的教学目标 E0 3.1.3 ▪ EU 3.1.2: 重新编写了关于抽象级别的段落 ▪ EU 3.4: 所有不需要在基础级应用的模型类型都被移到新的 3. 4. 6 节中 ▪ EU 3.6: 将标题更新为"需求文档和文档结构" EU 4 ▪ EU 4.1: 对利益相关者和利益相关者角色的描述更加准确。 ▪ EU 4.2: 对设计和创意生成技术的介绍和论证，已经更新且更加精确。 ▪ EU 4.3: 更新标题为"根据需求解决冲突" EU 6 ▪ 对 E06. 3. 1 和 E06. 5. 2 稍作修改
3.1.1	2024 年 01 月 1 日	更新企业设计

版本	日期	备注
3.2.0	2024 年 2 月 26 日	新企业设计的版面修正

目录

1 需求工程的简介与概述（L2）

目标： 了解什么是需求工程以及理解需求工程的价值

课时： 1小时

术语： 需求，需求规格说明，需求工程（RE），利益相关者，系统，需求工程师

学习目标

EO 1.1.1 牢记基本术语(L1)
EO 1.1.2 理解不同种类的需求(L2)
EO 1.2.1 解释需求工程的价值(L2)
EO 1.2.2 列举不充分的需求工程的症状。(L1)
EO 1.3.1 知道在哪里可以应用需求工程，以及在哪里出现需求。 (L1)
EO 1.4.1 知道需求工程的主要任务，并且为了执行它们必须要对需求工程的过程进行定制 (L1)
EO 1.5.1 描述需求工程师的角色和任务（L1）
EO 1.6.1 牢记需求工程师需要学什么（L1）

1.1 需求工程：是什么（L1）

人们和组织有对构建新事物或进化现有事物的愿望和需要。我们称这种需要为*需求*。

需要被构建或改进的事物可能是：

- 提供给客户的*产品*
- 提供给客户的*服务*
- 任何其他*可交付成果*，如可帮助人员和组织实现特定目标的设备、程序或工具
- 产品、服务或其他可交付成果的*构成*或*组件*

这些事物都可以被认为是*系统*。在本教学大纲中，我们使用术语*系统*来表示*利益相关者*要求的各种事物。*利益相关者*是影响系统需求或受系统影响的人员或组织。

需求工程的目标是说明和管理系统的需求，以便实施和部署的系统能满足利益相关者的愿望和需要。

在需求工程中，我们区分三种类型的需求[Glin2020]：

- *功能性需求*涉及由系统功能提供的结果或行为。这包括对数据或系统与其环境交互的需求。

- *质量需求*是关于未被功能性需求涵盖的质量关注点，如性能、可用性、安全性或可靠性。
- *约束*是一种需求，它限制解决方案在那些必要的、满足给定的功能需求和质量需求之外的空间。

1.2 需求工程：为什么 （L2）

充分的需求工程在开发和演进系统的过程中增加了*价值*：

- 降低开发错误系统的风险
- 更好地理解问题
- 估算开发工作量和成本的基础
- 测试系统的先决条件

不充分需求工程的典型症状是缺失的、不清楚的或不正确的需求。这尤其是由于：

- 直接进入构建系统阶段
- 与参与各方沟通的问题
- 假设这些需求是显而易见的
- 不充分的需求工程教育和技能

1.3 需求工程：在哪里 （L2）

需求工程可被应用于任何类型系统的需求。然而，需求工程目前的主要应用案例是由软件在其中发挥主要作用的系统来表示的。此类系统通常由软件组件、物理元素和组织元素组成。

需求可以以下列方式出现：

- *系统需求* — 系统应该做什么
- *利益相关者需求* — 从利益相关者的视角，他们想要什么
- *用户需求* — 从用户的视角，他们想要什么
- *领域需求* — 必需的领域属性
- *业务需求* — 组织的业务目标、目的和需要

1.4 需求工程：怎么做（L1）

需求工程中的主要任务是获取（4.2）、文档化（3）、确认（4.4）和需求管理（6）。工具支持（7）可能有助于执行这些任务。需求分析和需求冲突解决（4.3）被视为获取的一部分。为了正确执行需求工程活动，必须从广阔的可能性中（5）定制合适的需求工程过程。

1.5 需求工程师的角色和任务 （L1）

需求工程师通常不是一个职称，而是人们在其中扮演的一个*角色*：

- 获取、记录、确认和/或管理需求是他们职责的一部分；
- 对需求工程有深入的了解；
- 能够缩小问题和潜在解决方案之间的差距。

实际上，业务分析师、应用程序专家、产品所有者、系统工程师，甚至开发人员都在扮演需求工程师的角色。

1.6 从需求工程中学习什么（L1）

本教学大纲涵盖需求工程师必须学习的基础技能集合。它包括：需求工程（2）的基本原则、如何以各种形式记录需求（3）、如何用各种实践详细阐述需求（4）、如何定义和使用适当的需求工程过程（5）、如何管理现有的需求（6）以及如何采用工具支持（7）。

2 需求工程的基本原则（L2）

目标： 了解和理解需求工程的原则

课时: 1小时30分钟

术语： 上下文，需求，需求工程（RE），利益相关者，共识，确认

学习目标

EO 2.1.1 列举需求工程的原则（L1）
EO 2.2.1 牢记与原则相关的术语（L1）
EO 2.2.2 解释原则和为什么它们重要的原因（L2）

2.1 原则概述（L1）

需求工程受一组基本原则的控制，这些原则适用于需求工程中的所有任务、活动和实践。以下九项原则是本教学大纲后续各章介绍的实践的基础。

1. 价值导向：需求是达到目的的手段，而不是目的本身
2. 利益相关者：需求工程是满足利益相关者的愿望和需要
3. 共 识：没有共同基础，成功开发系统是不可能的
4. 上下文：系统不能孤立地被理解
5. 问题-需求-解决方案：不可避免交织在一起的三个层次
6. 确认：未确认的需求毫无用处
7. 进化：不断变化的需求不是偶然的，而是正常情况
8. 创新：做更多同样的事是不够的
9. 系统地、有纪律地工作：没有需求工程我们不能工作

2.2 原则的解释（L2）

原则 1 – 价值导向：需求是达到目的的手段，而不是目的本身。

需求的价值等于其收益减去获取、记录、确认和管理需求的成本。需求的收益是它有助于：

- 构建满足利益相关者愿望和需要的系统；
- 在开发系统时降低失败风险和高昂的返工成本。

原则 2 – 利益相关者：需求工程是满足利益相关者的愿望和需要

需求工程是理解利益相关者的愿望和需要，因此正确处理利益相关者是需求工程的核心任务。每个利益相关者在要构建的系统上下文中都扮演一个角色，例如：用户、顾客、客户、操作员或监管机构。利益相关者也可能同时具有多个角色。对于具有太多个人的利益相关者角色或当个人未知时，可以用被称为*角色*的虚构原型描述来替代。仅仅考虑最终用户或顾客的要求是不够的。这样做意味着我们可能会错过其他利益相关者的关键需求。能够提供系统使用时反馈的用户也应被视为利益相关者。

利益相关者可能有不同的需要和观点，这可能会导致需求冲突。识别和解决此类冲突是需求工程的任务。

让正确的人员参与利益相关者的角色对于成功的需求工程至关重要。4 中介绍了识别利益相关者、确定利益相关者优先级以及与利益相关者一起工作的实践。

原则 3 – 共识：没有共同基础，成功开发系统是不可能的

需求工程创建、促进和确保相关方之间的共识，相关方包括：利益相关者、需求工程师和开发人员。有两种形式的共识：

- *显性的共识*（通过文档和商定的需求达成）
- *隐性的共识*（基于对于需要、愿景、上下文等的共识）

领域知识、以前的成功协作、共享文化和价值观以及相互信任是实现共识的推动力，而地理距离、外包或高流动率的大型团队则是障碍。

实现共识的成熟做法包括创建术语表（3.5）、创建原型（3.7）或使用现有系统作为参考点。评估共识的实践包括预期结果的示例、探索原型或估算实施需求的成本。减少误解的影响最重要的做法是使用具有短反馈回路的过程（5）。

原则 4 – 上下文：系统不能孤立地被理解

系统被嵌入到*上下文*中。如果不理解该上下文，就不可能正确描述系统。在需求工程中，系统的上下文是系统环境的一部分，它与理解该系统及其要求相关。*系统边界*是一个系统与其周围上下文之间的边界。最初，系统边界通常不清楚，它甚至可能随着时间的推移而改变。

澄清系统边界，并定义系统与其交互的上下文元素之间的外部接口，是需求工程真正的任务。同时系统的*范围*，即开发系统时可以塑造和设计的事物的范围，需要被确定。我们还需要考虑所谓的*上下文边界*，该边界将系统环境中和需求工程相关部分与剩余其他部分分开。

当描述系统时，仅考虑系统边界内的需求是不够的。在需求工程中，还必须考虑：

- 上下文中可能会影响系统需求的变更；
- 与系统相关的现实世界的需求（以及如何将它们映射到系统需求）；
- 使系统正常运行，并满足实际需求所必须具备的上下文假设。

原则 5 – 问题-需求-解决方案：不可避免相互交织的三个层次

当利益相关者对现状不满意时就会出现*问题*。为了处理问题或简化问题，*需求*要捕获利益相关者需要的东西。满足这些需求的社会技术系统就构成一种*解决方案*。

问题、需求和解决方案不必按顺序依次出现。解决方案的想法可能会创建用户需要，这些需要必须作为需求进行制定并在实际解决方案中实现。这是典型的创新时的情况。

- 问题、需求和解决方案紧密交织在一起：它们不能孤立地被处理。
- 然而，需求工程师的目标是在思考、沟通和记录时尽可能将问题、需求和解决方案彼此分开。这种关注点的分离使需求工程的任务更易于处理。

原则 6 – 确认：未确认的需求毫无用处

最终，我们必须确认被部署的系统满足利益相关者的愿望和要求。为了从一开始就控制利益相关者不满意的风险，必须在需求工程期间就开始确认需求。我们必须检查是否：

- 利益相关者之间就需求达成了协议；
- 需求充分满足利益相关者的愿望和需要；
- 上下文假设（见上方原则 4）是合理的。

在 EU 4.4 中讨论了确认需求的实践。

原则 7 – 进化：不断变化的需求不是偶然的，而是正常情况。

系统及其需求不断*进化*。这意味着它们会不断变化。系统更改一条需求或一系列需求的请求可能是由以下原因引起的：

- 业务流程变更
- 竞争对手推出新产品或服务
- 客户改变其优先级或意见
- 技术上的变化
- 法律或法规的变更
- 来自系统用户的反馈，要求新功能或更改功能

此外，需求可能会因为利益相关者在验证需求时的反馈而改变，由于发现了以前所提出的需求中的错误，或者因为利益相关者的需求改变而改变。

因此，需求工程师必须追求两个看似矛盾的目标：

- 允许需求变化
- 保持需求稳定

6.7 中讨论了如何实现这一目标的细节。

原则 8 – 创新：做更多同样的事是不够的

准确地向利益相关者提供他们想要的东西，就会错失构建能更好地满足利益相关者需求的系统的机会。好的需求工程不仅是努力满足利益相关者，还要让他们感到快乐、兴奋或感到安全。这正是创新的最终目的。

需求工程塑造创新系统：

- 在小范围内，力求令人兴奋的新功能和易用性；
- 在大范围内，力求颠覆性的新想法。

其中 4.2 讨论了促进需求工程创新的几种技术。

原则 9 – 系统地、有纪律地工作：没有需求工程我们不能工作

有必要采用合适的过程和实践来系统地获取、记录、确认和管理需求，无论实际使用的开发流程如何。即使一个系统是以临时方式开发的，系统化和规范化的需求工程也能提高最终系统的质量。

需求工程中没有可以在每一种给定情况下都有效或至少在大多数情况下有效的单一的过程或实践。系统化、有纪律地工作意味着需求工程师：

- 根据给定的问题、上下文和环境，调整他们的流程和实践；
- 不要总是使用相同的过程和系列实践；

- 不要在没有反思的情况下重复使用过去成功的需求工程工作中的过程和实践。

对于每一项需求工程意图、流程、实践和制品，必须选择最适合的特定情况。详情在3、4、5 和6中详细阐述。

3 制品和文档实践（L3）

目标：理解需求工程中制品的基本作用并创建制品

课时： 7 小时

术语：

制品，基于自然语言的制品，基于模板的制品，基于模型的制品，术语表，质量标准，需求规格说明

学习目标

EO 3.1.1 了解需求工程制品的特征并列举常用的制品类型（L1）
EO 3.1.2 了解每个制品的用途并了解制品的使用期限（L1）
EO 3.1.3 解释需求的不同抽象级别，包括如何选择适当的抽象级别和细化程度（L2）
EO 3.1.4 了解制品中需要考虑的方面以及这些方面之间的相互关系（L1）
EO 3.1.5 命名通用文档指南（L1）
EO 3.1.6 描述为什么对将要使用的制品进行计划是值得的（L1）
EO 3.2.1 了解基于自然语言的制品以及它们的优点和缺点（L1）
EO 3.2.2 解释编写好的基于自然语言的需求的规则（L2）
EO 3.3.1 了解基于模板的制品的类别以及它们的优点和缺点（L1）
EO 3.3.2 详细说明使用句型模板的个体需求和用户故事（L3）
EO 3.3.3 详细说明使用表单模板的用户故事（L3）
EO 3.4.1 理解需求工程中模型的角色、目的和使用（L2）
EO 3.4.2 理解需求工程中建模的优点和局限性（L2）
EO 3.4.3 了解术语模型，建模语言，活动模型，活动图，类模型，类图，上下文模型，上下文图，领域模型，目标模型，交互作用模型，过程模型，时序图，状态图，状态机，状态机图，用例，用例图（L1）
EO 3.4.4 理解如何在既定情景为指定需求选择一个合适的模型类型（L2）
EO 3.4.5 理解并说明在适当的地方用UML编写的以下类型的简单模型：上下文模型，用例和用例图，领域模型，类模型，活动模型，过程模型，以及状态图（L2）
EO 3.4.6 使用UML类图详细说明一个系统的数据或领域中对象的简单模型（L3）
EO 3.4.7 通过UML活动图详细说明一个简单的系统功能或业务过程（L3）
EO 3.5.1 解释术语表的目的以及如何创建术语表（L2）
EO 3.6.1 了解常用的需求规格说明文档（L1）
EO 3.6.2 解释哪些文档结构服务于哪些目的以及构建的标准（L2）
EO 3.7.1 了解不同种类的原型以及它们的用途（L1）
EO 3.8.1 了解单一需求的质量标准（L1）
EO 3.8.2 了解制品的质量标准（L1）

3.1 需求工程中的制品（L2）

制品是工作过程中产生的被记录的中间或最终结果。需求工程中有各种各样的制品，例如，从短暂的图形式的草图到演进的用户故事集合，再到正式发布合同要求的数百页规格说明文档。

3.1.1 制品的特征（L1）

制品的特点是其目的、表现形式、大小和试用期限。以下制品对于给定的目的在实践中频繁出现。注意，一个制品可能包含其他制品。

- 单一需求的制品包括个体需求和用户故事。
- 一组连贯需求的制品包括用例，某种类型的图形模型 (3.4), 任务描述，外部接口描述，以及史诗。
- 需求文档结构的制品包括系统需求规格，产品和冲刺待办列表，以及故事地图。
- 其他制品包括术语表、文本注释、图形式草图以及原型。

制品可以用各种形式*表现*：

- 基于自然语言的（3.2）
- 基于模板的（3.3）
- 基于模型的（3.4）
- 其他表现形式，如图纸或原型（3.7）

大部分的制品在数据库或需求工程工具中以电子方式*存储*为文件。非正式的、临时的制品也可能被存储在其他媒介——例如，纸张或看板上的便利贴。

在看制品的使用期限时，我们将其区分为三个类别：

- *临时的制品*：被创建是为了支持沟通以及建立共识。
- *进化的制品*：随着时间的推移在几个迭代中发展；需要一些元数据 (6.5); 可能需要变更控制。
- *持久的制品*：已经*基线化*或*发布*；需要一整套元数据(6.5)；；必须遵循变更过程 (6.3, 6.4)。

一个临时的制品可以被转换成一个进化的制品（通过维持临时的制品并增添元数据）。类似地，一个临时的或者不断进化的制品可以通过基线化或发布而变成持久的制品。

3.1.2 抽象级别 （L2）

需求通常存在于许多不同的*抽象级别*上——例如，从一个新业务过程的高级需求具体到非常详细的需求，比如特定软件组件对异常事件的反应。

合适的抽象级别的选择取决于需求描述的主题以及规格说明的目的。然而，不要混合不同抽象级别的需求，这一点是重要的。在小型和中型的制品中，需求应该或多或少处于相同的抽象级别上。

在大型的制品，比如系统需求规格说明中，不同抽象级别的需求应该通过相应地构建规格说明书来保持独立(3.6)。一个高度抽象的需求可能会被细化为更低、更具体的几个详细需求。

当在持久的制品（比如业务需求规格说明、利益相关者需求规格说明或愿景文档）中表示高等级的业务需求或利益相关者需求时，它们会先于系统需求规格说明。在其他设置中，业务需求、利益相关者需求以及系统需求可能共同发展。

3.1.3 细化程度（L2）

具体需求的*细化*程度取决于几个因素，特别是：

- 问题和开发上下文
- 对问题的共识程度
- 留给设计师和程序员的自由程度
- 设计与实施过程中利益相关者的快速反馈的可用性
- 详细的规格说明的成本与价值
- 强制实施的标准和监管约束

指定的需求细化程度越高，最终得到意外的或未指定的事物的风险就越低。然而，规格说明的成本会随着细化程度的增加而增加。

3.1.4 制品中需要考虑的方面（L1）

当在制品中指定需求时，需要考虑不同的方面。

1. 需求按其种类（1.1）分为：

a) 功能性需求

b) 质量需求

c) 约束

2. 功能性需求侧重于系统功能的不同方面：

a) 结构和数据

b) 功能和流程

c) 状态和行为

3. 最后，需求只能在系统上下文中被理解 (原则 4 in 2):

a) *系统上下文*，包括外部行为者

b) *系统边界*和外部接口

上述方面之间存在许多相互关系和依赖关系。例如，用户（上下文）发出的请求可能触发状态转换（状态和行为），该状态转换发起一个行动，然后是另一个行动（功能和流程），该行动需要数据（结构和数据），以便在给定的时间间隔（质量）内向用户（上下文）提供结果。

一些制品侧重于一个特定的方面，并从其他方面进行了抽象。尤其是对于需求模型（3.4）的案例。其他制品，比如系统需求规格说明，涵盖了所有这些方面。当不同的方面被记录在单独的制品中，或者在同一个制品的不同章节中，这些制品或章节之间必须保持一致。

3.1.5 通用文档指南（L1）

与所使用的技术无关，在创建制品时适用以下指南：

- 选择符合*预期目的*的制品；
- 通过引用内容来*避免冗余*，而不是重复相同的内容；
- *确保*制品之间*没有不一致*，特别是当它们涵盖不同的方面时；
- *应一致使用*术语表中定义的*术语*。
- 适当地*构建制品*。

3.1.6 计划将要使用的制品（L1）

每个项目设置以及每个领域都是不同的，所以必须为每个行为定义一组作为结果的制品。因此，以下问题必须达成一致：

- 需求应该被记录在哪些制品中以及为了什么目的？
- 需要考虑哪些抽象级别？
- 在每个抽象级别，需求必须被记录到哪一个细化程度？
- 需求应该如何在这些制品中表示？

应该在项目的早期定义将要使用的制品。这样做有如下几个优点：

- 有助于计划工作量和资源；
- 确保使用合适的符号；
- 确保所有结果都记录在正确的制品中。
- 确保不必进行信息的重大重组和"最终编辑"。
- 有助于避免冗余，从而使得工作量更少以及易维护性更好。

3.2 基于自然语言的制品（L2）

自系统化的需求工程成立以来，自然语言需求一直是实践中指定需求的核心手段。

基于自然语言的制品有以下主要优点：

- 不受约束的自然语言具有极强的表现力和灵活性；
- 几乎任何可以想象到的需求在任何方面都可以用自然语言表达；
- 自然语言在日常生活和学校教学中使用，所以阅读和理解自然语言中的文本不需要专项训练。

这些优点带来的代价是，用自然语言编写的文本经常能够以不同的方式解释，这在说明需求时构成了一个问题。此外，在这些文本中，检测歧义、遗漏以及不一致是困难且代价高昂的。

编写良好的基于自然语言的需求可以由以下方式支持：

- 书写简短且结构良好的句子；
- 定义并一致使用统一的术语（3.5）。
- 避免含糊的或引起歧义的术语和短语；
- 了解下面列出的技术写作的陷阱。

在用自然语言编写技术文档时，有一些众所周知的陷阱应该避免或小心使用（[GoRu2003]）。

需要避免的事物是：

- 不完整的描述
- 未指明的名词
- 不完整的条件
- 不完整的比较

需要小心使用的事物是：

- 被动语态
- 全称量词（比如"所有"或"从未"）
- 名词化（即由动词派生的名词，例如："鉴定"）

3.3 基于模板的制品（L3）

基于模板的制品被用来克服基于自然语言的制品的一些不足之处，通过为需求提供预先定义的结构。

- *句型模板*为表达需求的短语提供了预定义语法结构，特别是个体需求或用户故事。
- *表单模板*在表单中提供了一组需要填写的预定义字段，例如，用于编写用例或可测量的质量需求。
- *文档模板*为需求文档提供一个预定义结构。

文献中描述了各种各样的模板。[ISO29148]、[MWHN2009]和[Rupp2014]为个体需求提供了句型模板。[Cohn2004]为用户故事定义了广泛使用的句型模板，[Cock2001]为用例描述了表单模板。

[Laue2002]为任务描述提出了一个模板。[ISO29148]和[RoRo2012]为全部规格说明提供了文档模板。此外，客户可能会在项目中规定客户专用模板的使用。

基于模板的制品的优点：

- 提供了一个清晰的、可重复使用的结构；
- 有助于捕获最相关的信息；
- 使需求和需求规格说明统一；
- 提高需求和需求规格说明的整体质量。

基于模板的制品的缺点和陷阱：

- 人们经常关注模板的格式完成度，而不是内容；
- 没有被包含在模板中的方面更有可能被遗漏。

3.4 基于模型的制品（L3）

用自然语言表示的需求有局限性[Davi1993]，特别是在获得一组需求概述以及理解需求之间的关系方面。建模需求解决了这些限制。

3.4.1 需求工程中模型的角色（L2）

*模型*是对现实中现有的一部分或将要创造的现实的一部分的抽象表现。现实的概念包括任何可以想象的一组元素、现象或思想，包括其他模型。就模型而言，建模的现实部分被称为*原型*。软件工程领域之外的模型的例子是建筑信息模型（BIM） [ISO19650], 它为规划、建造和管理建筑和其他建筑元素所需的元素建模。

在需求工程中，模型有助于理解需求之间的关系和相互连接，并提供一组需求的概述。这主要是通过关注某些方面来实现的（例如行为），同时从所有其他方面进行抽象。使用模型的图形符号也可以支持获得概述。然而，模型同样可以通过非图形的方式表现，例如使用表格。

与用自然语言表示的需求相比，需求模型具有以下优势：

- 使用图形模型比用自然语言指定需求之间的关系和相互连接更容易理解；
- 侧重于单一方面减少了理解建模需求的认知负荷；

- 需求建模语言有限制句法，减少了可能的歧义和遗漏。

模型也有局限性：

- 维持侧重于不同方面的模型之间的一致性是具有挑战性的；
- 为了连贯的因果理解，需要整合来自不同模型的信息；
- 模型主要侧重于功能需求；大多数质量需求和约束不能通过合理的工作量在模型中表达出来；
- 图形建模语言的限制句法意味着不是每个相关的信息项都可以在模型中表达。

因此，需求模型和自然语言中的需求经常被结合在一起。

在需求工程中，模型可以被用来：

- 作为替代以文本形式表达的需求的一种手段，*描述*部分甚至完全的（主要是功能性的）需求；
- 将一个复杂的现实*分解*成定义明确并互补的方面；每个方面都由一个特定的模型表示；
- *转述*以文本形式表达的需求，以便提高它们的可理解性，特别是关于它们之间的关系；
- *确认*以文本形式表达的需求，以发现遗漏、歧义和不一致之处。

*建模语言*用于表达模型。一些建模语言，例如[OMG2017]或[OMG2013]，已经标准化。当使用非标准建模语言指定需求时，需要一个图例来解释所使用的建模语言的句法和语义。

有许多类型的模型可以在需求工程中使用。需求工程师必须理解在给定的情况下哪种模型类型最适合描述需求。

在早期阶段，需求工程师通常从建模上下文(3.4.2)或预期系统的目标开始。

3.4.2 建模上下文（L2）

侧重于上下文方面的模型描述了系统在其环境中的结构嵌入，以及系统与系统上下文中的角色之间的交互。

*上下文模型*描述一个系统以及系统上下文中与系统交互的角色。上下文模型还描绘了系统和它的上下文之间的接口（例如，就交换了什么信息而言）。

*上下文图*被用作表示上下文模型的图形建模语言。上下文图没有标准化的符号。来自结构化分析[DeMa1978]的上下文图或定制的框线图[Glin2019]可以用以表达上下文模型。

在建模语言 UML 中 [OMG2017]，*用例图*提供了，根据系统的用例和通过这些用例与系统交互的系统上下文中的角色，对系统及其上下文建模的方法。

*用例*从角色的角度对系统上下文中的角色和系统之间的动态交互进行建模。用例大多是使用自然语言（3.3）的表单模板或使用 UML 活动图（3.4.4）编写的。

3.4.3 建模结构和数据（L3）

侧重于结构和数据方面的模型指定了系统或领域的静态结构属性的需求。

静态领域模型指定了在感兴趣的领域中的（业务）对象及其关系。它们可以用 UML 类图来表达 [OMG2017]。

*类模型*主要指定系统的类及它们的属性和关系。类代表现实世界中有形和无形的实体，系统必须了解它们才能履行任务。UML 类图是典型地被用来作为类模型的建模语言。

3.4.4 建模功能和流程（L3）

侧重于功能和流程方面的模型描述了从给定输入或执行（业务）流程必需的行动产生所需结果而必需的行动顺序的需求，包括行动之间的控制流和数据流以及谁负责哪个行动。

*活动模型*被用于指定系统功能。在建模语言 UML[OMG2017]中，*活动图*被用于表达活动模型。它们为建模行动以及行动之间的控制流提供了基础。活动图还可以表达谁负责哪个行动。高级建模元素（CPRE 基础级别未涵盖）提供了建模数据流的方法。

*过程模型*被用来描述业务过程或技术过程。它们可以用 UML 活动图或特定的过程建模语言来表达，比如 BPMN[OMG2013]。在 CPRE 基础级别，我们只使用 UML 活动图进行过程建模。

3.4.5 建模状态和行为（L2）

侧重于状态和行为的模型，根据对事件的状态相关反应或组件交互的动态，描述系统或领域组件的行为的需求。

*状态机*对触发从一种状态转换到另一种状态的事件以及发生状态转换时需要执行的行动进行建模。*状态图*[Hare1988]是状态按层次和/或正交方式分解的状态机。状态机，包括状态图，可以在建模语言 UML[OMG2017]中用*状态机图*（也被称为*状态图*）来表达。

3.4.6 需求工程中的进一步模型类型（L1）

在 CPRE 基础级，对模型的理解和应用仅限于选定的、重要的模型类型。在需求工程中，还有更多的模型类型被使用。在 CPRE 的基础级，了解它们以及它们的用途即可。

*目标模型*表示一组目标、子目标以及它们之间的关系。目标模型还可能包括实现目标所需的任务和资源、想要实现目标的参与者，以及阻碍实现目标的障碍。

在 SysML 中[OMG2019], *模块定义图*可以通过使用系统和角色的原型块来表达上下文图。模块定义图也可以用来为系统的概念实体和它们之间的关系建立系统的结构模型。

*领域故事模型*可用于对功能和流程进行建模，通过指定角色如何与领域中的设备、制品和其他实体交互的可视化故事，通常使用领域特定的符合 [HoSch2020]。它们是理解将要运行的系统中的应用领域的一种方法。

*交互模型*为对象或角色之间的动态交互建模。

UML *时序图*是描述对象之间交互的广泛使用的方法。

3.5 术语表（L2）

在每个不止一个人参与的需求工程意图中，都存在对术语缺乏共识的风险——也就是说，有些人以不同的方式解释同样的术语。为了缓解这个问题，相关术语的共识被记录在一个术语表中。

术语表是以下定义的中心集合：上下文特定的术语、在给定的上下文中具有特殊含义的日常术语、缩写和首字母缩略词。同义词（表示同一个事物的不同术语）应这样标记。同形同音异义词（对不同的事物使用同一个术语）应避免或这样标记。

术语表适用以下规则：

- 集中管理；
- 整个系统开发过程中维护；
- 界定某人或小组来负责术语表；
- 使用统一的风格和结构；
- 让利益相关者参与进来，并寻求有关术语的一致意见；
- 让所有相关人员都使用术语表；
- 强制使用术语表；
- 检查制品是否正确使用术语表。

3.6 需求文档和文档结构(L2)

需求规格说明文档（3.1.1）包括几个需求工程制品。因此，用定义明确的结构来组织这些文档很重要，以便于创建一致且易维护的需求集合。除了需求之外，需求文档还可能包含额外的信息和解释——例如：术语表、验收条件、项目信息或技术实现的特征。

需求也可以用传统文档以外的文档结构来组织。

常用的文档有：

- 利益相关者需求规格说明
- 用户需求规格说明（利益相关者需求规格说明的子集，只涵盖用户的需求）
- 系统需求规格说明
- 业务需求规格说明
- 愿景文档

常用的供选择的文档结构有：

- 产品待办列表
- 冲刺待办列表
- 故事地图

文档结构的选择和所选结构的内部组织都取决于：

- 选定的开发过程（5）
- 开发类型和领域
- 合同（客户可以规定使用给定的文档结构）
- 文档的大小

文档模板可以帮助构建需求规格说明。模板可以在文献[Vole2020]、[RoRo2012]和标准[ISO29148]中获得。模板也可以从以前的类似项目中重复使用，或者由客户强制实行。组织也可以决定创建一个模板作为内部标准。

3.7 需求工程中的原型（L1）

在需求工程中，*原型*是一种通过实例描述需求以及确认需求的手段。尤其是，如果所涉及的利益相关者不想编写和评审基于自然语言的、基于模板的或基于模型的制品，那么可以使用原型。

探索型原型[LiSZ1994]用于在不同级别的保真度上创建共识、阐明需求或者确认需求。它们在使用后被丢弃。

- *线框图*是用简单的材料或草图工具创建的低保真原型，主要用于讨论并确认设计思路和用户界面概念。
- *Mock-up 模型*是中保真原型。当描述数字系统时，它们使用真实的屏幕和点击流程，但没有真正的功能性。它们主要用于描述并确认用户界面。
- *原生原型*是实现系统关键部分的高保真原型，在一定程度上利益相关者可以使用原型来观察原型化的系统部分是否将如预期地运作和表现。

根据保真度程度的不同，探索型原型可能是一个昂贵的制品，因此始终在成本和获得的价值之间权衡。

进化型模型[LiSZ1994]是形成待开发系统核心的试点系统。最终系统是通过在多次迭代中增量扩展和改进试点系统进化而来的。

3.8 制品和需求的质量标准（L1）

一个需求需要满足特定的质量标准才能被认为是好的需求。在以价值为导向的现代需求工程方法中（2 中的原则 1），满足质量标准的程度应与这个需求所创造的价值相对应。这意味着需求不必完全遵守所有的质量标准——但是需求的价值越高，就越和质量标准相关，以降低失败的风险。

充分性和可理解性是单一需求最重要的质量标准。没有它们，不管需求是否满足了所有其他的标准，该需求都是无用的，甚至是有害的。

*单一需求*的质量标准：

- 充分的（描述真实的、达成一致的利益相关者的需要）
- 必要的
- 明确的
- 完整的（自包含的）
- 可理解的
- 可验证的

正如3.1.1 中所描述的，需求通常记录在涵盖单一或多重需求的各种制品中。以上质量标准应该用于在制品中创建良好的单一需求。对于覆盖不止一个需求的制品，应该额外地考虑以下质量标准。

覆盖*多重需求*的制品质量标准：

- 一致的
- 非冗余的
- 完整的（没有遗漏已知的和相关的需求）
- 可变更的
- 可追踪的
- 符合的

4 需求疏理的实践（L3）

目标：　理解识别需求来源，获取需求，识别并解决冲突，以及确认需求实践的使用

课时：　4小时30分钟

术语：需求来源，系统边界，系统上下文，需求获取，需求确认，利益相关者，卡诺模型，冲突

学习目标

EO 4.1.1　确定系统的边界以关注相关需求（L3）
EO 4.1.2　牢记要创建的系统的相关来源（L1）
EO 4.1.3　识别利益相关者并编写利益相关者列表（L3）
EO 4.1.4　理解利益相关者管理的好处（L2）
EO 4.2.1　理解卡诺模型如何能够帮助获取正确的需求（L2）
EO 4.2.2　理解收集技术与设计和创意生成技术之间的不同（L2）
EO 4.2.3　理解如何在特定情况下选择合适的获取技术（L2）
EO 4.3.1　牢记冲突的不同类型（L1）
EO 4.3.2　理解解决冲突所需要的活动（L2）
EO 4.3.3　理解如何应用适当的冲突解决技术（L2）
EO 4.4.1　理解为什么需求文档需要被确认（L2）
EO 4.4.2　牢记需求确认的四个重要方面（L1）
EO 4.4.3　理解如何应用适当的需求确认技术（L2）

4.1 需求的来源（L3）

需求的质量和完整性在很大程度上取决于所涉及的需求来源。缺少相关的来源将导致对需求的理解不完整，或者导致不完整的需求。需求来源的识别是一个需要不断重新考虑的迭代和递归过程。

对将要开发的系统上下文的共识（2 中的原则 3）是能够识别相关需求来源的先决条件。在系统边界和上下文边界之间的区域被称为（系统）上下文（2 中的原则 4）。需要（系统）上下文来理解将要开发的需求的本质，从而识别需求的原始来源。

需求来源被分为三种类型：

- 利益相关者
- 文档
- 系统

系统的利益相关者（参见[Glin2020]中的定义；也可以参见2 的原则 2）是主要的需求来源。典型的利益相关者角色包括 [BiSp2003]:

- 用户（也称终端用户）
- 赞助商
- 管理人员
- 开发人员
- 官方
- 客户

此外，*受系统影响*的个人或组织也应被视为（间接）利益相关者。

对利益相关者的系统的识别，应该发生在开发项目的开始阶段，并且在整个开发过程中将结果作为持续的活动进行管理。这包括确定利益相关者的角色和担任这些角色的人。

对于所有具有用户界面的系统，*系统的终端用户*组成了需求工程师特别感兴趣的利益相关者组。终端用户应被汇总成组（例如，按类似的角色、任务或责任）。

当可以单独确定终端用户时，应选择每个组的代表。否则，可以定义角色来代表相关的终端用户群体 [Coop2004]。

识别相关的利益相关者和利益相关者角色的潜在来源有：

- 典型的利益相关者组和角色的清单
- 组织的结构
- 业务过程文档
- 市场报告
- 为识别*额外*利益相关者的初始利益相关者

利益相关者应该被记录在最新的利益相关者列表中，（至少）包含以下信息：

- 名称
- 功能（角色）
- 其他个人和联系资料
- 项目进展期间的时间和空间可用性
- 关联
- 专业知识的领域和范围

- 就项目而言的目标和兴趣

如果利益相关者的权利和义务不明确，或者利益相关者的需要没有得到充分解决，就可能出现利益相关者问题。利益相关者关系管理[Bour2009] 是应对利益相关者问题的有效途径。

在大多数系统上下文中，有很多的来源可用。为了一个成功的新系统，它们也必须被考虑，因为大多数利益相关者不会谈论显而易见的事：他们的"潜意识"需求（4.2）。

需求的额外来源包括：

- 现存和遗留系统
- 过程文档
- 法律或监管文档
- 公司特定的规程
- 关于潜在的未来用户的（市场）信息

需求的另一个来源可以通过观察完全不同领域中的类似情况被发现。

4.2 需求的获取（L2）

在获取中，需求工程师的任务是理解利益相关者的愿望和需要，同时确保通过应用适当的技术来获取所有相关需求来源的需求已经被收集。获取的一个主要观点是将隐性的要求、希望和期望转化为显性需求。

为了获取需求，了解每个需求的本质和价值是至关重要的。这些可能会随着项目的不同以及时间的推移而改变。卡诺模型[KaeA1984]将需求分为三个相关的类别：

- 喜悦因素（同义词：兴奋因素、无意识的需求）
- 满意因素（同义词：绩效因素、有意识的需求）
- 不满意因素（同义词：基本因素、潜意识的需求）

有许多不同的技术可以获取这些需求类别。我们区分为：

- 收集技术

- 设计和创意生成技术

收集技术是针对需求获取的已确立的技术[BaCC2015]，它通过调查不同的来源，帮助获取满意因素和不满意因素。

主要分为四类：

- 提问技术
- 协作技术
- 观察技术
- 基于制品的技术

*设计和生产创意的技巧*是为了在需求获取期间激发创造力。他们的目的是创造解决问题的想法，并探索设计理念 [Kuma2013]。这可能会导致新的或创新的要求，这些需求通常是令人愉快的。这类技术的流行例子是头脑风暴[Osbo1979]、类比、原型设计（例如模拟）场景和故事板。

与设计和想法产生相关的一个更加广泛的概念是*设计思维*。这里存在着不同的方法，例如 *d.school* [Sdsc2012] 和 *Designing for Growth* [LiOg2011], 他们提供了可以用于获取需求的大量技术。

获取技术应该能够检测出各种各样的需求——功能和质量需求，以及约束等。在实践中，质量需求和约束往往很少得到关注。

为了获取*质量需求*，类似于 ISO25010 标准[ISO25010]的质量模型应该被用来作为检查单。这个模型在量化需求方面也很有帮助。

*可以*通过考虑潜在解决方案空间的可能的限制来发现*约束*——例如，技术、法律、组织、文化或环境问题。

选择正确的获取技术是一个关键的核心竞争力，它取决于许多不同的因素，例如：

- 系统类型
- 软件开发生命周期模型
- 有关人员
- 机构设置

最好的结果通常是结合不同的获取技术。[CaDJ2014]代表了一种选择技术的系统方法。

4.3 根据需求解决冲突〔L2〕

获取技术本身并不能确保需求集的结果作为一个整体是一致的、完整的、符合的，等等。(3.8). 然而，对于最终的集合，所有利益相关者必须理解并同意与他们相关的所有需求。如果某些利益相关者不同意，这种情况就被认为是一种冲突，应该得到相应的解决。应该根据冲突的类型和上下文信息选择合适的冲突解决技术。这需要深入理解需求冲突的本质以及利益相关者的态度。

识别和解决冲突的任务是：

- 识别冲突
- 分析冲突
- 解决冲突
- 解决冲突的文件（已做出的决定）

区分不同的冲突类型是有用的[Moor2014]。以下类型的冲突通常需要需求工程师的关注：

- 主题冲突
- 数据冲突
- 利益冲突
- 价值冲突
- 关系冲突
- 结构冲突

要成功地解决冲突，可以运用以下常见技巧：

- 认同
- 折中
- 投票
- 否决
- 定义不同的变体

此外，还有一些辅助技术，例如：

- 考虑所有事实

- 三面思考法（Plus-Minus Interesting）
- 决策矩阵

4.4 需求确认（L2）

确认需求是迈向成功系统的重要步骤（2 中的原则 6）。预先确保需求质量将减少下游工作的浪费。确认需求意味着检查手头的制品以及其中个体需求的质量（详见3.8）。

在需求确认中需要考虑的重要方面是：

- 正确的利益相关者参与
- 将缺陷的识别与修复分开
- 从不同的角度来确认
- 反复确认

有几种确认技术（例如[GiGr1993]，[OleA2018]）。这些确认技术通常分为：

- *评审技术*，包括：
  - 走查
  - 审查
- *探索技术*，例如：
  - 原型制作
  - Alpha 测试和 beta 测试
  - A/B 测试[KoTh2017]
  - 构建最小可行性产品（MVP）
- *样品开发*

这些技术在形式上和工作量上有所不同。选择哪种技术取决于诸如软件开发生命周期模型、开发过程的成熟度、系统复杂程度和风险水平、任何法律或监管要求和/或需要审计跟踪等因素。

5 过程及工作结构（L3）

目标：解释需求工程过程的概念并应用适当的过程配置

课时：　　1小时15分钟

术语：　　过程，需求工程过程

学习目标

EO 5.1.1　了解影响需求工程的重要因素（L1）
EO 5.1.2　理解这些因素是如何以及为什么会产生影响（L2）
EO 5.2.1　理解在配置需求工程过程时要考虑的方面（L2）
EO 5.3.1　了解典型的需求工程过程配置（L1）
EO 5.3.2　理解需求工程过程的配置步骤（L2）
EO 5.3.3　对于简单的系统和开发设置，选择并应用一个合适的需求工程过程配置（L3）

塑造和构建在给定的上下文中完成需求工程工作的过程是必需的。由于没有一种万能的需求工程过程（1.4），因此必须配置适合给定开发和系统上下文的、量身定制的需求工程过程。

需求工程过程塑造了不同参与者（例如，客户、用户、需求工程师、开发人员、测试人员）之间的信息流和通信模式，并且还定义了将要使用或生产的制品。因此，需求工程过程提供了获取、记录、确认和管理的框架。

5.1 影响因素（L2）

很多因素会影响需求工程过程的配置。主要因素有：

- 整个过程适合：需求工程过程必须适合整个系统开发过程。
- 开发上下文
- 利益相关者的能力和可用性
- 共识
- 被开发系统的复杂性和关键性
- 约束
- 可用的时间和预算
- 波动的需求
- 需求工程师的经验

对影响因素的分析为如何配置需求工程过程提供了信息。影响因素也限制了合理工程配置的空间。例如，当利益相关者只在项目开始可用时，工程就不能选择建立在持续的利益相关者反馈之上。

5.2 需求工程过程方面（L2）

在配置需求工程过程时，需要考虑三个决定性因素[Glin2019]。

时间方面：线性和迭代

在一个线性的过程中，需求会在过程的单个阶段预先被指定。在一个迭代的过程中，需求是增量地被指定，从一般目标和一些初始的需求开始，然后在每次迭代中添加或修改需求。

选择*线性*需求工程过程的标准：

- 系统的开发过程是计划驱动的，大部分是线性的，或者选择线性的需求工程过程。
- 利益相关者了解他们的需求，并且可以预先指定它们。
- 需求需要一份全面的需求规格，作为外包系统的设计和实现的合同基础。
- 在开发早期阶段，监管机构需要一个全面的、正式发布的需求规格说明。

选择*迭代*需求工程过程的标准：

- 系统开发过程是迭代的、敏捷的；
- 许多需求并不是预先知道的，但是会在系统开发的过程中出现并发展；
- 利益相关者是可以利用的，这样可以建立短反馈回路，以降低开发错误系统的风险；
- 开发的持续时间允许不止一个或两个迭代；
- 轻松更改需求的能力是很重要的。

目的方面：规范性和探索性

在一个规范的需求工程过程中，需求规格说明构成了一个契约：所有的需求都是有约束的，并且必须实现。在探索需求工程过程中，只有目标是预先知道的，而具体的需求是需要被探索的。

选择*规范的*需求工程过程的标准：

- 客户需要一个系统开发的固定合同；

- 功能和范围优先于成本和期限；
- 指定系统的开发可通过招标或外包的形式进行。

选择*探索性*需求工程过程的标准：

- 利益相关者最初对他们的需求只有一个模糊的概念；
- 利益相关者积极参与并提供持续的反馈；
- 截止日期和成本优先于功能和范围；
- 不是最开始就清楚哪些需求实际上应该被实现，以及它们将按照什么顺序被执行。

目标方面：以客户为中心还是以市场为导向

在面向客户的需求工程过程中，系统由客户订购并由供应商开发。在市场导向的需求工程过程中，系统是为市场而开发的一种产品或服务，以特定的用户群体为目标。

选择*面向用户的*需求工程过程的标准：

- 系统将主要由订购该系统并为其开发支付费用的组织使用；
- 重要的利益相关者主要与客户的组织联系在一起；
- 个人可以被认为是利益相关者角色；
- 客户想要一个需求规格说明来充当一份合同。

选择*市场导向的*需求工程过程的标准：

- 开发组织打算将系统作为产品或服务在某些细分市场中销售；
- 潜在用户不能被单个识别；
- 需求工程师必须设计需求，使其符合目标用户的预期需要；
- 产品负责人、市场人员、数字设计师和系统架构师是主要的利益相关者。

提示和说明

- 上文提出的标准是启发性的而不是固定的规则。例如，系统的开发外包，最好是用一个规范性的需求工程过程来完成，而不是用一个探索性的过程，因为客户和供应商之间的合同通常是基于一个全面的需求规格说明。但是，也可以基于探索性的需求工程过程协商外包合同。

- 线性的需求工程过程只有在复杂的需求变更流程到位的情况下才有效。
- 线性的需求工程过程意味着很长反馈回路：为了减少开发错误系统的风险，必须对需求进行充分的确认。
- 在定义需求工程过程时，通常会同时选择*线性*和*规范*。
- 探索性的需求工程过程通常也是迭代的过程（反之一样）。
- 在市场导向的过程中，用户的反馈是确认产品是否真正满足目标用户需求的唯一手段。
- 市场导向方面往往与线性和规范方面不能很好地结合。

5.3 配置需求工程过程（L3）

在一个具体的系统开发上下文中，负责需求工程的人员必须配置要应用的需求工程过程。在分析影响因素的基础上 5.1,5.2 中描述的过程方面适当的组合可以被采用[Glin2019]。下面，我们描述三种典型的组合。

参与式需求工程过程：迭代的、探索性的、面向客户的

主要应用案例：	供应商与客户紧密合作；利益相关者强烈地参与到需求工程和开发过程中
典型的制品：	带有用户故事和/或任务描述、原型的产品待办清单
典型的信息流：	利益相关者、产品负责人、需求工程师和开发人员之间的持续互动；可能包括来自用户的反馈

合同化的需求工程过程：典型有线性的（有时是迭代的）&规定性的&面向客户的

主要应用案例：	在需求分析阶段之后，需求规格说明构成了一个系统开发的契约基础，人员没有参加到规格说明并且很少与利益相关者交互
典型的制品：	典型的系统需求规格，由基于自然语言的需求和模型组成

典型的信息流：	主要是从利益相关者到需求工程师

产品导向的需求工程过程： 迭代的&探索的&和市场导向的

主要应用案例：	组织设定并开发软件，目的是将其作为产品或服务进行销售或分发
典型的制品：	产品待办清单、原型
典型的信息流：	产品负责人、市场营销、需求工程师、数字设计师、开发人员之间的交互，以及（可能）来自客户/用户的快速反馈

注意，在系统和开发上下文中，上述配置可能都不适用。例如，监督管理的约束可能会强制使用符合给定标准的过程，如 ISO/IEC/IEEE 29148[ISO29148]。

在配置需求工程过程时，我们建议使用 5 个步骤：

1. 分析影响因素 (5.1)
2. 评估标准方面 (5.2)
3. 配置过程 (5.3)
4. 确定制品 (3)
5. 选择合适的实践

6 需求管理实践 （L2）

目标： 理解需求管理的需要和好处

课时： 2 小时

术语： 需求管理，变更管理，可追踪性，需求属性，需求生命周期，优先级处理

学习目标

EO 6.1.1 了解需求管理是关于什么以及为什么它是被需要的（L1）
EO 6.2.1 解释为什么需求制品需要一个状态/生命周期模型（L2）
EO 6.3.1 在特定的项目情况下，解释需求版本的概念是什么 (L2)
EO 6.4.1 了解需求配置和基线使用（L1）
EO 6.5.1 了解需求的目标属性（L1）
EO 6.5.2 解释在给定的项目情况下，合适的需求属性集是什么样的。(L2)
EO 6.5.3 解释视图的目的，并命名需求的不同视图（L2）
EO 6.6.1 说明需求可追踪性的原因（L1）
EO 6.6.2 总结隐性和显性可追踪性之间的区别（L1）
EO 6.6.3 了解显性可追踪性如何能够被文档化（L1）
EO 6.7.1 了解如何处理线性（基于计划的）和敏捷方法的变化（L1）
EO 6.8.1 了解优先级处理的原因并了解有意义的评估标准 (L1)
EO 6.8.2 列出步骤来对需求进行优先级排序（L1）
EO 6.8.3 列出不同类别的优先级处理技术（L1）

6.1 什么是需求管理? [L1] （L1）

需求管理是管理记录在各种制品中的现有需求的过程 特别是，这包含了存储、更改和跟踪需求[Glin2020]。需求管理可以以不同的方式在不同的级别上发生，这取决于所选择的开发过程和上下文 – 例如，参见[Leff2011]，[Rupp2014]，[WiBe2013]。无论何种情况，需求管理的任务，都是以一种使项目中的所有角色能够更高效、更有效率地工作的方式来维护需求。

6.2 生命周期管理（L2）

生命周期管理是指根据其生命周期中的状态跟踪所有制品的过程。每个文档化的需求和每个制品都有自己的生命周期：在对其进行评审、返工、合并、商定等之前，它是被创建、评估和完善的。为了识别哪个制品处于哪种状态，需要一个生命周期模型来定义每个被允许的生命周期状态和状态转换。一个制品的真实状态应该总是清晰的，（通常）包括它的转换历史。

6.3 版本控制（L2）

需求的版本控制，是指在演进中保持对所有制品的跟踪的过程。制品中的任何变更都应该通过一个新版本来反映。版本控制允许将制品的历史追溯到它的起源，并将制品恢复到任何更早的版本。为此，版本控制需要有三项措施：

- 用于唯一标识制品版本的版本号。
- 改动的历史记录。
- 制品存储的概念。

必须考虑所有制品的版本控制[WiBe2013]。版本号通常至少由两部分组成：版本和增量。

6.4 配置和基线（L1）

需求*配置*是一组包含需求的一致的制品。每个配置都是为特定目的定义的，并且最多包含每个制品的一个版本[Glin2020]。例如，配置的目的是评审一组制品或者促进对开发工作的评估。

*基线*是制品一个稳定的，变更控制的配置，用于项目中的发布计划或其他交付里程碑 [Glin2020]。

配置具有以下属性：

1. 逻辑连接
2. 一致性
3. 独特性
4. 不变性
5. 重置依据

6.5 属性和视图（L2）

*属性*是被需要用来记录制品的重要元数据并且通常用于回答项目或产品生命周期中的许多问题。

需求属性的目的是使团队和利益相关者能够在项目期间的任何时刻获得关于他们所需的需求信息。

定义相关的属性集取决于项目中不同利益相关者的信息需要。现有的标准，例如[ISO29148]，提供了最具相关性的属性的概述。

*视图*是从全部的需求集中摘录出来，这些需求仅包含当前感兴趣的内容。从技术角度来看，视图是过滤器和分类设置的组合，其他参与者可以通过存储选定的组合来使用和重用这些设置。

我们区分三种类型的视图

- *选择视图*
- *投影视图*
- *聚合视图*

在大多数情况下，需求视图是用于创建报告的选择性、投影和聚合视图的组合。

6.6 可追踪性（L1）

可追踪性[GoFi1994]是将需求追踪到其*起源*（例如，利益相关者、文档、证明等等）并且*转发到后续的制品（例如测试用例）*，以及该需求所依赖的其他需求。

可追踪性是需求管理的先决条件，并且通常是标准、法律和法规显性要求的。实现可追踪性本质上意味着维护不同工作制品之间的依赖关系(0)。在不同的抽象层次 (3.1.2)，不同的细节层次(3.1.3)，以及出于分析、遵从性和信息的原因，维护他们与前置任务和后置任务之间的关系。

可追踪性可以被隐性地记录下来，通过结构化和标准化的制品，或者显性地记录下来，通过基于不同形式的唯一标识符将制品相互关联[HuJD2011]。常用的表达形式有超链接、引用、矩阵、表格或图表。

6.7 处理变化（L1）

需求不是静态的。需求的变更是由于许多不同的原因而发生的，并且需要适当地处理（2 中的原则 7），例如通过创建一个正式的变更请求，或者通过向产品待办事项列表中添加一个新项。

变更的决策制定、计划和控制实现取决于开发方法和变更发生的时间点。

在*线性*的方法中，变更决策通常由变更控制委员会（在项目中）或变更咨询委员会（在操作中）做出。在多*迭代*的方法中，产品负责人将变更包含在产品待办列表中，并相应地为新的项设定优先级。

6.8 优先级处理（L1）

并不是所有的需求都同等重要 [Davi2005]。评估和优先级处理被用于确定下个产品发布或增量的最相关的需求。

需求*评估*是确定优先级处理的基础，通常通过使用多种评估标准来确定，如业务价值、紧急程度、工作量、依赖关系等。

一个需求的优先级描述了单一需求相对于其他特定标准需求的重要性[Glin2020]。*优先级*本身是基于一个或多个标准来执行的；这主要取决于所选择的优先级处理技术。

优先级处理的步骤：

- 为优先级处理确定主要目标和约束条件
- 定义预期的评估准则
- 定义必须参与的利益相关者
- 定义必须定义优先级的需求
- 选择优先级处理技术
- 执行优先级处理

*优先级处理技术*可划分为：

- *随机*优先级处理技术
- *分析型*优先级处理技术

7 工具支持（L2）

目标： 提供实现需求工程工具和方面的角色概述

课时： 30 分钟

术语： 工具，需求工程工具

学习目标

EO 7.1.1 了解不同类型的需求工程工具（L1）
EO 7.2.1 解释引入需求工程工具时需要考虑什么（L2）

7.1 需求工程中的工具 （L1）

需求工程过程可以被支持专门任务和活动的工具支持。由于需求工程过程是非常独立的 (5)，现有的需求工程工具通常只关注需求工程中的某些方面，很少支持所有的活动。选择工具之前，需求工程师应该决定在需求工程过程中应该支持哪些任务和活动以及如何支持。我们区分了工具支持的功能：

- 需求的管理：
  - 定义和存储需求属性
  - 定义需求优先级
  - 管理版本和配置
  - 可追溯和跟踪需求
  - 管理需求变更
- 需求工程过程的管理：
  - 测量和报告需求工程过程
  - 测量和报告产品质量
  - 管理需求工程的工作流程
- 关于需求知识的文档化：
  - 共享需求
  - 创建对需求的共识
- 需求的建模
- 需求工程中的协作

- 需求的测试/仿真

工具通常混合了上述的特性。为了确保适当地覆盖所有的需求工程任务，可以组合不同的工具。

有时，其他种类的工具，例如 Office 或问题跟踪工具，被用来记录或管理需求。这些工具有局限性，并且应该只在需求工程过程处于控制之中，以及需求是达成一致的并且相当稳定时使用。

7.2 工具介绍（L2）

选择需求工程工具与选择其他用途的工具没有什么不同。在成功选择之前，必须先描述目标、上下文和需求[Fugg1993]。

只有在介绍了需求工程流程和技术后，才能寻求合适的工具。工具介绍需要明确需求工程的职责和规程。在介绍需求工程工具的过程中，有以下几个方面是相关的：

- 考虑许可证成本之外的全生命周期成本
- 考虑必需的资源
- 利用试点项目去规避风险
- 按照已定义的标准去评估工具
- 指导员工如何使用工具

参考文献

[AnPC1994] Annie I. Antón, W. Michael McCracken, Colin Potts: Goal Decomposition and Scenario Analysis in Business Process Reengineering. CAiSE (Conference on Advanced Information Systems Engineering), 1994, 94–104.

[BaCC2015] K. Baxter, C. Courage, K. Caine: Understanding Your Users: A Practical Guide to User Research Methods, 2nd edition. Morgan Kaufmann, Burlington, 2015.

[BiSp2003] K. Bittner, I. Spence: Use Case Modelling. Pearson Education, Boston, 2003.

[Bour2009] L. Bourne: Stakeholder Relationship Management: A Maturity Model for Organisational Implementation. Gower Publishing Ltd, Birlington, 2009.

[CaDJ2014] D. Carrizo, O. Dieste, N. Juristo: Systematizing requirements elicitation technique selection. Information and Software Technology 2014, 56(6): 644–669.

[Cock2001] A. Cockburn: Writing Effective Use Cases. Addison-Wesley, Boston 2001.

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

[Coop2004] A. Cooper: The Inmates Are Running the Asylum: Why High-Tech Products Drive Us Crazy and How to Restore the Sanity. Que, Indianapolis, 2004.

[Davi2005] A. M. Davis: Just Enough Requirements Management – Where Software Development Meets Marketing. Dorset House Publishing, New York, 2005.

[Davi1993] A. M. Davis: Software Requirements – Objects, Functions, & States, 2nd edition, Prentice Hall, New Jersey, 1993.

[DeMa1978] T. DeMarco: Structured Analysis and System Specification. Yourdon Press, New York, 1978.

[Fugg1993] A. Fuggetta: A classification of CASE technology. IEEE Computer 1993, 26 (12): 25–38.

[GiGr1993] T. Gilb, D. Graham: Software Inspection. Addison Wesley, Boston, 1993.

[Glin2019] M. Glinz: Requirements Engineering I. Course Notes, University of Zurich, 2019. https://www.ifi.uzh.ch/en/rerg/courses/archives/hs19/re-i.html#resources. Last visited July 2020.

[Glin2020] M. Glinz: A Glossary of Requirements Engineering Terminology. Version 2.0. https://www.ireb.org/en/downloads/#cpre-glossary. Last visited July 2020.

[GoFi1994] O. Gotel, A. Finkelstein: An Analysis of the Requirements Traceability Problem. 1st International Conference on Requirements Engineering, Colorado Springs, 1994. 94–101.

[GoRu2003] R. Goetz, C. Rupp: Psychotherapy for System Requirements. 2nd IEEE International Conference on Cognitive Informatics (ICCI'03), London, 2003. 75–80.

[GRL2020] Goal oriented Requirement Language. University of Toronto, Canada https://www.cs.toronto.edu/km/GRL. Last visited May 2020.

[Hare1988] D. Harel: On Visual Formalisms. Communications of the ACM 1988, 31 (5): 514–530.

[HoSch2020] S. Hofer, H. Schwentner: Domain Storytelling — A Collaborative Modeling Method. Available from Leanpub, http://leanpub.com/domainstorytelling. Last visited July 2020.

[HuJD2011] E. Hull, K. Jackson, and J. Dick: Requirements Engineering. Springer, 3rd Ed, 2011.

[ISO29148] ISO/IEC/IEEE 29148: Systems and Software Engineering – Life Cycle Processes – Requirements Engineering, International Organization for Standardization, 2018.

[ISO19650] ISO 19650: Organization and Digitization of Information about Buildings and Civil Engineering Works, including Building Information Modelling (BIM)– Information Management Using Building Information Modelling – Part 1 and 2, International Organization for Standardization, 2018.

[ISO25010] ISO/IEC/IEEE25010:2011: Systems and software Quality Requirements and Evaluation (SQuaRE) –– System and software quality models. International Organization for Standardization, Geneva, 2011.

[Jack1995] M. A. Jackson: Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices. Addison-Wesley, New York, 1995.

[Jack1995b] M. Jackson: The World and the Machine. 17th International Conference on Software Engineering 1995 (ICSE 1995). 287–292.

[KaeA1984] N. Kano et al.: Attractive quality and must-be quality. Journal of the Japanese Society for Quality Control 1984, 14(2): 39–48. (in Japanese)

[KoTh2017] R. Kohavi, S. Thomke: The Surprising Power of Online Experiments – Getting the most out of A/B and other controlled tests. Harvard Business Review, Sept–Oct 2017: 74–82.

[Kuma2013] V. Kumar: 101 Design Methods – A Structured Approach for Driving Innovation in Your Organization. John Wiley & Sons, Hoboken, 2013.

[Laue2002] S. Lauesen: Software Requirements. Styles and Techniques. Addison-Wesley, Harlow, 2002.

[Leff2011] D. Leffingwell: Agile Software Requirements, Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison-Wesley, Boston, 2011.

[LiOg2011] J. Liedtka, T. Ogilvie: Designing for Growth: A Design Thinking Tool Kit For Managers. Columbia University Press, 2011.

[LiSZ1994] H. Lichter, M. Schneider-Hufschmidt, H. Zullighoven: Prototyping in Industrial Software Projects – Bridging the Gap Between Theory and Practice. IEEE Transactions on Software Engineering 1994, 20 (11): 825–832.

[MFeA2019] D. Méndez Fernández, X. Franch, N. Seyff, M. Felderer, M. Glinz, M. Kalinowski, A. Volgelsang, S. Wagner, S. Bühne, K. Lauenroth: Do We Preach What We Practice? Investigating the Practical Relevance of Requirements Engineering Syllabi – The IREB Case. CIbSE 2019: 476–487.

[Moor2014] C. W. Moore: The Mediation Process – Practical Strategies for Resolving Conflicts, 4th edition. John Wiley & Sons, Hoboken, 2014.

[MWHN2009] A. Mavin, P. Wilkinson, A. Harwood, and M. Novak: Easy Approach to Requirements Syntax (EARS). 17th IEEE International Requirements Engineering Conference (RE'09), Atlanta, Georgia, 2009. 317–322.

[OleA2018] K. Olsen et al.: Certified Tester, Foundation Level Syllabus – Version 2018. International Software Testing Qualifications Board, 2018.

[OMG2013] Object Management Group: Business Process Model and Notation (BPMN), version 2.0.2. OMG document formal/2013-12-09 http://www.omg.org/spec/BPMN. Last visited July 2020.

[OMG2017] Object Management Group: OMG Unified Modeling Language (OMG UML), version 2.5.1. OMG document formal/2017-12-05. https://www.omg.org/spec/UML/About-UML/. Last visited July 2020.

[OMG2019] Object Management Group: OMG Systems Modeling Language (OMG SysML™), Version 1.6. OMG Document formal/19-11-01. https://www.omg.org/spec/SysML/. Last visited January 2022.

[Osbo1979] A. F. Osborn: Applied Imagination, 3rd revised edition. Charles Scribner´s Sons, New York, 1979.

[RoRo2012] S. Robertson and J. Robertson: Mastering the Requirements Process, 3rd edition. Addison-Wesley, Boston, 2012.

[Rupp2014] C. Rupp: Requirements-Engineering und Management, 6. Auflage. Hanser, München, 2014. (in German).

[Sdsc2012] Stanford d.school: An Introduction to Design Thinking. Hasso Plattner Institute of Design, Stanford, 2012. https://dschool-old.stanford.edu/groups/designresources/wiki/36873. Last visited July 2020.

[vLam2009] Axel van Lamsweerde: Requirements Engineering: From System Goals to UML Models to Software Specifications. Chichester: John Wiley & Sons, 2009.

[Vole2020] Volere: Requirements Resources. https://www.volere.org. Last visited July 2020.

[WiBe2013] K. Wiegers and J. Beatty: Software Requirements, 3rd edition. Microsoft Press, Redmond, 2013.