EXIN Scrum Master & Product Owner 指南

作者：Johann Botha

书名: EXIN Scrum Master & Product Owner 指南

作者: Johann Botha

出版社: EXIN Holding BV

ISBN: 9789076531007 (eBook)

版本: 2021 年 9 月

译者: 马小莉、郑维亮、左茜、刘世超、吴昊、周琴、丁克斌、陈连生、黄灵

EXIN 国际信息科学考试学会特别感谢为本书中文版在翻译和审校方面作出特别贡献的 EXIN 志愿者。

目录

图表列表

简介

本书是 EXIN Agile Scrum Master（后续简称 ASM）和 EXIN Agile Scrum Product Owner（后续简称 ASPO）或者 EXIN Agile Scrum Product Owner Bridge（后续简称 ASPOB）认证体系的官方考试教材文献。

正因如此，本书的目标很明确，那就是帮助考生备考 ASM、ASPO 和/或 ASPOB 认证。

这些认证是***高级***认证，因此我们会假设您在阅读此书时，您对敏捷的基本概念和 Scrum 这种敏捷方法都已经有所了解。如果您希望了解基础知识，我们推荐您通读由 Nader Rad 和 Frank Turley 编著的 ***Agile Scrum Handbook*** 以及 Scrum.org 网站上的 ***Scrum Guide*** 2020（注：中文版为《Scrum 指南》2020 版）和 ***Nexus Guide***（注：中文版为《Nexus™ 指南》，最新版为 2021 年 4 月版）。

虽然 EXIN Agile Scrum Foundation（后续简称 ASF）不是本书范围内的认证的先决条件，但是我们强烈建议您取得该认证，除非您已经对 Scrum 的基本概念非常熟悉了。

同时我们还提供其他的 EXIN 敏捷认证，请访问 EXIN 中文网站获取最新信息：https://exinchina.cn/certification。

前言

我参与敏捷相关的工作可以追溯到很久远的时间——在大约 20 年前我使用另一种敏捷方法，即极限编程（Extreme Programming，下文简称 XP），来管理软件开发项目。当我带领一群特立独行的程序员为一家大型 IT 集团企业开启其互联网业务时，我们使用的就是 XP。从那时起，我在许多项目以及咨询和辅导活动中使用了 Lean、Scrum、ABC/DSDM、看板和 DevOps 方法和概念。

使用适应性的和经验性的方法（例如 Scrum）的自然结果就是，人们会根据自己的经验发展个人的偏好和偏见。我想要说的是，因为敏捷需要您基于多种方法来思考和行动，因此根据您自己的经验而由此得来的个人偏好是完全可行的。然而，我尝试不让我的个人偏见和偏好出现在本书中，但愿我能成功做到这一点。本书主要是关于 Scrum 的，我已尽我所能将本书的内容聚焦于此。

话虽如此，Scrum 常常得益于 Scrum 以外的方法和技巧。举两个简单的例子，我们会使用看板去可视化地管理迭代并且实现工作流，我们使用源自 ABC/DSDM 的 MoSCoW 作为一个可能的方法去对产品待办事项列表进行排序。

我希望您能喜欢这本书，因为这本书非常实用。该书不但可以作为考试辅导书使用，而且也可以作为 Scrum Master（后续简称 SM）、产品负责人（后续简称 PO）或者敏捷教练的参考指南。

真诚的，

Johann Botha，2020 年 11 月于约翰内斯堡。

1 设置敏捷场景

1.1 敏捷思维

大写字母开头的 Agile（名词）：涉及或者表示一种项目管理方法，通常用于软件开发。其特点是将工作任务拆分到若干个短周期工作阶段，并经常对计划进行重新评估与调整。如无意外，后续会将大些 A 开头的 Agile 翻译为“敏捷”。

小写字母开头的 agile（形容词）：能够快速轻松地移动；能够快速地思考、理解与响应。为了与前面的 Agile 做区分，后续不对其进行翻译，保留 agile 原文。

这些可以在澳大利亚 PM-partners 网站找到的简单定义，为我们完美地设置了敏捷场景。

1.1.1 敏捷作为一种“项目方法”

敏捷项目管理方法非常适合诸如软件设计和开发之类的经验性的项目，而它对于复杂系统中的任何项目也同样适用。这些项目中的变数与未知因素让传统的瀑布模式变得不切实际。

敏捷项目管理方法是一种强调在实验结果中发现概念、可接受在项目中演进交付的方法。

agile 的概念最早定义于 2001 年 2 月 11 日在犹他州山区的雪鸟滑雪胜地。敏捷软件开发宣言就是那次影响深远的会议的产物，如今它被称为敏捷宣言。

许多如今依然大受欢迎的敏捷方法的代表人物出席了这场历史性聚会。所有与会人员都确信，需要找一个当时使用文档驱动的重量级软件开发流程的替代品。

这 17 个先驱者提出敏捷宣言的原因包括：漫长交付周期的挫败感；耗费巨大精力去做的、结果被证明是无用的计划；以及项目阶段中持续在变化的需求，因为他们周围的世界在变化。这些挫败感造就了敏捷宣言及其十二项原则，它们解释了敏捷如何帮助组织以更快速和超乎想象的交付质量去交付更好的价值。

很快大家意识到，对于软件开发项目适用的东西，也同样适用于需求不明确的或者在持续演进的其他类型的项目。

如此程度的不确定性在项目中是普遍存在的。几乎所有的项目或多或少都必须在项目推进的过程中去面对模棱两可的、不断变化的需求并从错误中吸取经验。

也正因此，如果您处于一个未来不确定、交付物在项目的开始阶段无法被清晰定义的环境，并且您还希望在前进的过程中不断地学习，那么敏捷和敏捷方法将是您的唯一出路。

何出此言？因为在面对不确定性、模棱两可、以及在诸如组织和项目之类的复杂适应性系统（CAS）中时常发生的持续变化的需求时，经验性的方法会是最可行的方法。

em·pir·i·cal （ɛm-pîr-î-kl），经验性的，形容词——依赖或派生自观察或实验，或者能够以观察或实验的方式验证。它们是被实际经验指引的，而不是理论指引的。

让传统的（所谓的瀑布模式）项目管理如此受限的原因是假设我们可以制定详细的计划且应该先于项目执行。

大众似乎有一种错误的理解，即敏捷方法不做预先计划[1]；然而，事实却并非如此。很难发现一个敏捷项目是没有花些时间在预先规划[2]上面的。

敏捷方法较之前的区别是：您基于已知的东西去计划未知的东西，并且允许需要交付的人员在他们忙于交付的时候去回答问题。通常到了那个阶段，他们已从之前的项目阶段中学习到了很多东西，并且也理解事情需要运转的情景。往往随着项目的推进和逐渐展开，对项目所处情景的理解也会更好。

那些假设可以在项目开始阶段做出一个完美的、详细的计划的人，请注意：没有人足够聪明或幸运到能够计划和预测所有事情，尤其是在没有预先准备的情况下。

下一个我们需要提醒自己的事情是，*项目更多的是关于客户的预期，而不是需求规格说明书*。无论我们如何努力去尝试预先定义好需求规格说明书，我们都无法捕捉和量化客户和用户的所有预期。

有时候，客户、用户或者消费者需要看到一些实际的东西才能接受或拒绝它，因为它要么满足了他们的预期，要么不符合他们的预期。

用 Henry Ford 的话来说明这一点：如果我问我的客户他们想要什么，他们会说一匹更快的马。

我们常说，客户不在意您的输出（即您交付的东西）；他们在意的是结果（他们能通过您的交付物而可以做的事情或者能达成的目标）。这意味着，当项目的交付物（特定的输出），可以让客户达成他们想要或需要的目标（成果）时，客户就会满意了。

因此，价值通常由供应商和客户共同创造——这是值得铭记的东西。

请考虑如下问题：

您曾经参与过多少这样的项目，项目的交付物与客户认可的需求规格说明书非常接近，但是他们仍然不满意？

在我们开始以敏捷的方式做事之前，这个问题的答案可能是很多。

有两个原因可以在我们使用敏捷时提高客户满意度：

- 敏捷让用户和客户成为项目的一份子，这使得他们有机会尽早地提供频繁的反馈；
- 敏捷使我们有能力基于两周而非两年的预期去交付工作。

世界是会变的，我们客户的需求也一样。

---

[1] 计划，原文为 plan，名词，指的是计划的结果──译者注

[2] 规划，原文为 planning，名词，指的是做计划的过程。规划的最终产物是计划──译者注

1.1.2 敏捷作为一种思维模式和行为方式

值得一提的是，在 Snowbird 的那次会议中，最初的讨论大多都是围绕着一种不同以往的工作方式，以及怎么将 TQM[3]（全面质量管理，即计划、执行、检查、处理（PDCA）循环）和精益（即丰田生产系统）应用到软件开发中。

这也难怪当我们去看敏捷的十二条原则时，它们与精益的原则有着惊人的相似之处。

敏捷宣言作者的初衷并非是开发一种软件开发的方法。相反，他们的关注点是项目团队应当如何去重视特定的事物（A 高于 B），并且将这些价值观转化成一套可以应用到软件开发过程中的原则。

因为从业者需要将原则转换为实践，在这过程中，敏捷方法也就自然而然地被开发出来了，并且目前所有敏捷方法都将这十二条原则作为指引或者道德规范。

现在应该是时候提醒您关于敏捷团队重视的四个价值观以及十二条原则了。敏捷价值观。

价值观一：**个体和互动** 高于 *流程和工具*。

价值观二：**工作的软件**（或其他任何您的项目需要交付的东西）
高于 *详尽的文档*。

价值观三：**客户合作** 高于 *合同谈判*。

价值观四：**响应变化** 高于 *遵循计划*。

单单看着这些价值观，我们就能够清楚地明白，相比我们在项目中做的其他事情来说，帮助客户获取价值才是更重要的。请注意，这些宣言并非说流程、工具、文档、合同以及计划是不重要的。这些价值观只是阐述那些更加重要的事情。那些事情包括：与人们交谈和一起工作、给客户满足他们需求的东西、引导客户参与价值交付的过程、知道什么时候环境和需求已经发生变化。

下面列举的是基于上述价值观而构建的并使之具有可操作性的敏捷原则：

原则 1. **我们最重要的目标，是通过持续不断地及早交付有价值的软件使客户满意**当客户能够更快更频繁地接收到交付物并因此达成他们的目标时，他们会更满意，而不是需要在版本之间进行漫长的等待。

原则 2. **欣然面对需求变化，即使在开发后期也一样。**为了客户的竞争优势，敏捷过程掌控变化。随着客户环境和领域的频繁变化，客户的需求也在不断变化。这不是范围蔓延，这是现实。您只能接受现实，否则必将失败。您需要记住的是，每一个新的需求很可能都对应存在一个已经失效的旧需求。

原则 3. **经常地交付可工作的软件**，相隔几星期或一两个月，倾向于采取较短的周期。这与原则 1 接近，但并非完全相同的。在这里，我们说我们不仅应该更快地交付客户可以使用的东西，而且还应该以可预测的频率或者节奏进行交付（另请参见原则八）。

原则 4. **业务人员和开发人员必须相互合作，项目中的每一天都不例外。**尽早并且频繁地让干系人参与到项目中来。如果可以的话，您甚至可以尝试让他们参

[3] TQM，Total Quality Management，全面质量管理。

与到每一个交付迭代中去，而不仅仅是评审会。我们常常引导软件的用户参与到我们的 Scrum 团队中去。谁会比用它（软件）工作的人（即干系人）更了解需求呢?

原则 5. **激发个体的斗志，以他们为核心搭建项目。**提供所需的环境和支援，辅以信任，从而达成目标。这是给您的温馨提示，如果您想在组织内部不进行文化变革的情况下尝试和使用敏捷，那么很大概率您会失败。

原则 6. **不论团队内外，传递信息效果最好效率也最高的方式是面对面的交谈。**请记住，80%的交流都是非语言的！我们正在学习如何应对一个物理连接更少的世界，但是在开发团队里，面对面的互动才是最重要的。注意：在后疫情（Covid-19，2020 年新冠疫情）时代的世界里，以虚拟团队的方式来工作成为新的现实。但我们仍然推荐您尽可能地使用面对面的沟通方式。正因如此，虚拟团队的会议应该以视频会议的方式进行。

原则 7. **可工作的软件是进度的首要度量标准。**它能完成工作吗？它可以使用吗？它产生预期的结果了吗？如果不是，那么您失败了。

原则 8. **敏捷过程倡导可持续开发。**责任人、开发人员和用户要能够共同维持其步调稳定延续。再次重申，本原则与原则一和原则三密切相关。在敏捷项目中，整个项目不会有时间终点，但是团队正在做的事情需要有终点。随着业务所处环境的频繁变化，需要也会不断变化。这也意味着敏捷项目是永远不会结束!

原则 9. **坚持不懈地追求技术卓越和良好设计，敏捷能力由此增强。**这意味着确保团队成员拥有合适的技能、他们遵循良好设计的规则，并且不允许积累技术债务，这些是至关重要的。

原则 10. **以简洁为本，它是极力减少不必要工作量的艺术。**本原则也可称为金发姑娘原则；每一件事都需要是正确的。它旨在作为原则九的平衡，在原则九中我们关注质量和细节。但是，我们需要提醒自己，更多不一定更好。我们需要尽一切努力去使其发挥作用，不多也不少。

原则 11. **最好的架构、需求和设计出自自组织团队。**对于开始敏捷之旅的组织来说，此原则是最困难的，因为它暗示着管理风格、绩效考核方式的转变，有时甚至是组织架构的转变，包括角色和职责的重新定义。这里的重点是将工作的细节留给专家，也就是团队成员——因此不要进行微观管理。如今的多数员工是知识工作者，他们不需要监督；他们需要的是激励和支持。

原则 12. **团队定期地反思如何能提高成效，并依此调整自身的举止表现。**这意味团队有一定的成熟度，并且有意愿成为自己命运的主宰者。为了使本原则有效果，组织管理者必须解决组织变革的难题并且建立一个高度信任的环境。

至此，您应该明白为什么可以说 agile 是一种思维模式，它将以客户为中心的原则作为指引，而这原则可能是最重要的。“敏捷 agile”说的是发起一套有意识的文化上的、组织上的以及行为上的变革，此变革应该从组织中的领导层开始。

单纯使用本书中描述的技巧去交付功能、重命名角色或者实践相关仪式[4]，都不会让敏捷方法在您的组织内获得成功，除非您先变得思维敏捷，而这本身并非易事。如果我们仔细思量一下，为什这么多的敏捷计划失败了——那么可以肯定的回答是，您必须首先变得思维敏捷才能让敏捷方法在您的组织内获得成功。

[4] 在当前 Scrum Guide 2020 版本中，使用“事件”来代替“仪式”。

然而，固守一种似乎可行的模式或者方法并开始倡导这个方法是人的天性。这些方法都是敏捷方法。但是，如果您在您的组织中使用上述十二条原则，无论您使用的是何种方法以期实现业务敏捷，那么您都是在践行 agile。

您此时可能想问，既然如此本书还有何存在的必要性呢?

本书描述的敏捷与 Scrum 方法、职责与技术，它们是极其有价值的，只要您不成为它们的奴隶，它们将会为您的组织带来丰硕的成果。它们是有用的辅助工具，将会为您提供良好的服务；但是不要让任何敏捷方法来决定您做什么事以及如何做。那样的话，您就失去了真正的组织敏捷性。敏捷方法应该始终为敏捷思维服务，反之则不然。

由于本书实际上是围绕着如何最大程度地利用 Scrum 作为一种敏捷方法而展开。从现在开始，除了一些在特定的场景下使用的名词，比如“敏捷行为”（agile behavior）、“敏捷思维模式”（agile mindset）或者“敏捷原则”（agile principles），我们都会使用敏捷（Agile）这个词，而不是 agile。

1.2 敏捷案例

向人兜售敏捷的优点在最开始是相当困难的，因此最好从个人层面去做这件事。在这个简短的章节里，我们将向您展示一个可能的方法供您斟酌。事实已证明该方法是可行的。

本方法采取的是交互式的故事/角色扮演方式，故事中有怀疑论者和拥护者两种角色。该故事是一个 IT 项目的故事，但只要修改一下场景和一些参数，它可以轻易地适用于任何环境。由于多数人见过线上项目的成功，也见过其失败，所以它似乎是一个合适的例子。场景的一些背景信息：

XYZ 公司是一家在快速消费品市场中蓬勃发展的零售商。在过去 10 年间，他们取得了成功，并将零售业务从 2 家门店拓展到了 15 家，门店遍布全国最大的 10 个都市。

近来，XYZ 公司逐渐感受到来自线上零售商的压力，尽管这家公司也拥有广泛的线上营销业务，管理层意识到他们必须迅速提升线上营销能力，例如网店。如果他们能够成功做到这一点，他们就可以利用现在的市场地位和关系同新的竞争者

为了达到预期目标，如下列举了两个选项供参考。

选项 1——传统瀑布模型的项目方法

- 调研竞争对手在做什么 （2 个月）
  - 由 IT 人员完成
- 调研可用的技术栈 （2 个月）
  - 由 IT 人员完成
- 定义线上零售功能的详细规格书并创建业务案例 （4 个月）
  - 业务分析师（IT 人员）收集业务的需求
- 设计一个线上零售功能 （4 个月）
  - 由 IT 人员完成
- 开发该线上零售功能 （12 个月）
  - 由各个 IT 部门分别完成各自的工作，包括开发、测试、发布管理、IT 运营
- **总工期** **24 个月**

业务有关的问题

1. 您是否认为上述内容是您业务中此类项目的一个合理的代表?
   - 答案通常是：**是。**
2. 在上述的竞争环境下，您可以为一个解决方案等待两年的时间吗?
   - 答案通常是：**否。**
3. 如果您必须要等待两年，那将会发生什么?
   - 答案通常是显而易见的，那将会前途渺茫。
4. 为了能与新加入者竞争，您认为多长时间是一个合理的时间?
   - 答案通常是 4 到 6 个月.
5. 您是否认为在上述场景中第 5 个月定义完成并达成一致的需求，将会与项目预期完成的 19 个月后的需求保持一致?
   - 答案通常是：**否。**
6. 您是否认为项目开始的第 8 个月提交审批的业务案例，在 16 个月后仍然有效?
   - 答案通常是：**否。**
7. 从您以往的经验来看，此类项目是否可以在预算内如期完成?
   - 答案通常是：**否。**
8. 您可以接受哪些妥协?
   - 这个问题的答案不尽相同，但是总体而言会出现如下选项，（选项顺序无关紧要）：
     a. 在较短的时间内提供功能较少的产品
     b. 将商店外包给另一方
     c. 利用现有的平台
     d. 花更多的钱将产品尽快完成

虽然选项 b 和 c 是可行的且应该加以探讨，但是他们都存在明显的缺点——即失去对客户的控制权和较少的利润。如果公司现金流充足，花更多的钱也是可行的。但是经验表明，这个选项通常是不可行的，或者至少是不受欢迎的。

选项 2——敏捷项目方法

根据组织自己的需求和竞争者在做的事情来识别高层次的需求
（1 个月）

由业务完成，IT 人员从旁协助

识别一两个最有可能在线上可以大卖的产品系列　（2 周）

由业务完成，IT 人员从旁协助

线上识别和调查支撑这些产品所需的最小功能集　（2 周迭代设计）

由 IT 人员完成，业务从旁协助

构建基本的购物车功能　（4 周迭代）

由 IT 人员完成

构建和测试用户界面　（4 周迭代）

由 IT 人员完成

优化解决方案　（2 周迭代）

由 IT 人员完成

发布基础功能

获得线上功能的总时间　（4 个半月）

增强线上商店的功能，最终将所有被认为可以在线上销售的产品可以在下一年销售，
（采用 4 周迭代的方式）

由 IT 人员完成，基于从业务处收集到的买家/客户行为的反馈

获得优化后的且功能齐全的在线功能　**18 个月左右**

<u>对业务的提问</u>

1. 线上购物功能发布的那一天，需求并未得到全部满足，您是否满意?
   - 答案通常是：**否。**
2. 如果所有的需求得到满足意味着需要再多等 19 个月，您是否愿意?
   - 答案通常是：**否。**
3. 您是否认为需求、或者您对需求的理解，会在发布后的 18 个月里面改变?
   - 答案通常是：**是。**
4. 对于常规的两年周期的项目，是否所有预先定义好的需求都被证实是必须的或者是有效的?
   - 答案通常是：**否。**

请回想一下您过去项目中的经历，经过项目定义和项目交付之间的两年时间之后：

5. 您是否请求过系统更新，为什么?
   - 答案通常是：**是**——因为出现了新的需求，市场、客户、竞争状态在一直变化等。
6. 在这两年期间，多少个新的需求被提出?

- 答案可能随行业不同而有所区别，通常是：**相当多。**

7. 现有的特性功能，有多少个您认为是至关重要的?
   - 答案可能随行业不同而有所区别，通常是：**大部分，但并非全部。**
8. 如上列举的两个场景，如果您是 XYZ 公司的管理者，您会选择哪一个选项?
   - 多数情况下，答案是**选项 2。**

这就引出了下图，为了实用，此图已做简化。时间，如同金钱一样，也是价值的。如果您比较一下价值/时间之间的关系方程式，你就会发现采用敏捷是理所当然的。

敏捷拥有可以适用于每个组织的潜能。有些组织在变革组织思维模式和工作方式的时候会比其他的组织面临更多的挑战，但是每个组织都可以的。那些拒绝变革的组织，很可能无法在这个飞速发展的数字时代多活一天。

但是请注意，从文化角度来说，敏捷不会在所有的环境都起作用，**除非高层管理人员至少有一些意愿去拥抱敏捷**，否则最好不要追求敏捷。

图 1：敏捷相比常规的项目管理方法更快地创造更多的价值

图片由 EXIN 基于以下内容创建：Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value.* Amazon Digital Services LLC – Kdp. Print us.

高层管理人员需要问自己的问题是：如果我们不去拥抱业务敏捷的概念，我们是否可以生存？回答"是"暗含着面对现实的意愿。敏捷转型并非易事，也不是一个容易的决策；它需要魄力和勇气去贯彻和执行。幸运总是眷顾勇敢的人，这在敏捷转型过程中，同样适用。

1.3 正确实现敏捷的关键成功因素

敏捷是一种使用较短的迭代周期向客户交付价值的方法。该方法中客户深度参与研发过程，跨职能团队负责交付成果的质量。如果再加入更多的描述，就要开始深入研究某个具体的敏捷方法了。

一般来说，不管您使用的是什么敏捷方法，都可以定义一些关键的成功因素。

1.4 领导力和行为、文化、道德以及信任

尽管任何组织或者组织中的一部分都可以使用敏捷方法，但是如果可以通过领导层营造合适的环境，敏捷可以发挥其最好的功效。

跟精益一样，敏捷首先是一种文化和行为现象，从本质上来说，如果组织的领导者理解并且致力于在组织中推行它们，那么它们推崇的这些行为变革将会最有效。这些文化上的变革很大程度上受到组织的领导风格的影响。

在过去的 20 年间，虽然人们时常说起仆人式领导，但管理者的行为方式、处事风格和管理风格却极少改变。如果问什么事情在组织中成功推行敏捷的过程中最重要，那就是组织中所有层级的管理者的态度转变。

之所以这么说，是因为敏捷项目是由那些自治的、自管理的、跨职能的团队在管理。为了让此事可行，管理和领导风格必须发生根本性的转变。

用 Michael Shota 的话来说就是：管理者必须明白，这与采用敏捷方法（Doing Agile）无关，我们要的是敏捷状态（Being Agile）。而敏捷状态（Being Agile）首先得从组织的领导者开始。

将敏捷成功落地的组织中的领导者会对员工如何执行任务有着强烈的兴趣，并且他们会以教练、老师与赋能者，而非管理者的角色，来启发员工去完成工作。

成功的敏捷实施需要**信任**和**开放**，意味着管理者对所有员工坚定不移地付出，并有着强烈的意愿让组织中的所有人变成他们所能成为的最好的人。这同样也要求沟通是**公开、坦率和公平的**，员工在与组织的领导层互动时可以互相信任。

当被问：及如何创造一个信任和开放的环境时，我们会说，管理者必须学会相信他们的员工有能力做正确的事情。

除非员工知道他们可以信任管理者，并且管理者不会因为他们尝试新事物、说出真相和坦诚相告而惩罚他们，否则他们将永远不会展现出可以让敏捷和精益正常运转所需的开放与坦诚的沟通。

Ernest Hemmingway 将其定义为：找出您是否可以信任某人的最好方法是信任他们。

下面的例子展示了为什么高信任度的环境比低信任度的环境对组织是更有利的：

低信任度

管理者不相信员工可以做好他们的工作，所以他们规定好员工应该如何做好本职工作。管理者规定的方法通常是不切实际的、过时的或者繁琐的，而员工可以以快速获得输出的方式工作。此种场景的危险性在于，那些构成既定方法、流程或者程序的安全网与控制流程现在被搁置一边了。

如果出现什么问题，没人会承认是他们违背了操作流程。错误经常因为员工试图逃避违规操作的惩罚而被掩盖。

结果就是环境变的越来越不稳定，直到一场大灾难发生并揭露这一切。

作为根因分析的一部分，寻找和惩罚过错方的行动就开始了。

然而这种做法的问题在于，即使过错方被找到了并且也受到惩罚了，组织和管理者的损失也已经造成，相关的损失也已无可挽回。让人吃惊的是，这种有害的行为竟然是许多组织正在使用的工作方法。

Richard DiPilla 曾经说过：雇佣棋手却把他们当成棋子是不明智的。同样的，雇佣聪明人却让他们遵循愚蠢的规则也是毫无意义的。

现在，让我们开始探索在与上述相反的敏捷环境中预期的行为是怎样的。

高信任度

管理者雇佣专业人员与专家，并期望他们为了组织更好的明天能够发挥他们的潜能和智慧。员工以小团队的方式在一起工作，通常是 3 到 9 人。他们会被赋予一个目标，如何达成这个目标以及如何组织工作并在规定的时间范围内达成这个目标，也是由自己决定。

因为团队成员更接近工作，因此他们能更好地理解事情是如何实际运转的以及具体需要完成的工作是什么。

他们使用由管理层定义的启发式控制[5]（heuristic controls）来设计他们自己的工作，以确保在实现最佳兴能的同时将风险最小化。如果出现什么问题，团队会毫无保留地沟通并尝试找到问题的永久性解决方案，并确认该问题不会再出现。

管理者会扫清前进道路上的障碍并成为团队间的促进者和调停者，并以给出建议的方式来支持团队。

由于团队认识到错误是难以避免的并被接受的，但是持续存在的错误是不能被容忍的，因此问题会被持续解决且很少会发展成灾难性的失败。在此场景中，组织层面的学习浪潮持续存在，组织的绩效也得到持续的提升。

- 上述两种工作环境，您认为哪一种是最让人向往的？
- 您更愿意在哪里工作呢？

您可能从上面的例子也发现了，在敏捷组织中需要推行的一个重要变革就是成为一个**学习型的组织**，而这与改变组织文化与行为紧密相关。转型为一个高信任度的环境是必须的，只有在这这种的环境中，人们才会对解决问题无所畏惧。当他们看见问题的时候，他们会在问题成为组织的重大问题之前解决他们。

[5] 启发式控制，通常只定义在特定地景的行为准则，则不定义具体的行为。如果您知道如何去做事以及预期的结果是什么，在多数情况下您可以自行决定合适的行为。但是，如果控制层定义好了一个行为，无论此行为合适与否，它都将得到执行。

学习型组织的概念在 Peter Senge 所著的 *The Fifth Discipline*[6]*（1990）*一书中得到了推广，书中有一些非常值得我们摘抄出来的内容。Senge 提出了学习型组织的如下 5 个特征：

1. **系统性思维**，意味着每一个人应该将组织理解为一个系统。人们对这个系统理解得越好，他们学到的就会越多，结果就是组织也会变得越来越好。
2. 系统性思维也意味着个体对于**学习和自我精进**的承诺成为组织的规范。学习不再仅仅是知识获取，而是专注于能力建设和效率提升。个体学习是个人的责任，而不仅仅组织的责任，并且"自我精进"应该成为日常的一部分。
3. 人们所持有的假设（**心智模型**）也需要转变。只有当促进寻根究底和信任的开放文化替代了对抗的态度时，这些假设才会转变。
4. 发展**共同愿景**对于员工来说是一个必不可少的自我学习的动机，因为它创造了一个集体认同感，为学习提供专注和能量。践行共同愿景为组织内部的沟通与协作创造了一个有利于发展相互信任的环境，并且鼓励员工去分享他们自身的经验和观点，从而提升组织层面学习的效果。
5. **团队学习**是通过分享积累的个人知识来实现的。团队学习或者分享型学习的好处就是，员工可以更加快速成长，组织层面的解决问题的能力也因为更好的知识获取途径和专家的存在而得以提升。个体参与到对话和讨论中，并达成团队的共识。个体必须拥有切实的分享知识的途径。

1.4.1 敏捷以及它对组织架构的影响

理论上来说，敏捷适用于任何组织。如果做得好，敏捷将不可避免地开始改变组织的结构、影响其架构以及如何最好地定义角色与职责。

因为敏捷需要自治的、跨职能团队去执行项目，这将不可避免地导致更扁平化的组织以及需要更少监督的、多技能的、多才多艺的员工。

我们有很充分的理由去相信，未来的组织将就是那样的。随着工作场所中越来越多的工作者成为知识工作者，对复杂管理结构的要求也变得越来越不重要。

[6] 彼得·圣吉，《第五项修炼》，中信出版社，2009-10-1 出版，ISBN：9787508616827

2 实施敏捷

如今存在着许多的敏捷方法，Scrum 是其中最常用的，其他的一些主流的方法你可以在本书的附录 A 中找到。

精益是所有敏捷方法和技术的共同祖先。这些方法都有共同的基础元素，尽管本书的意图是专注于介绍敏捷和 Scrum，但是不时也将介绍其他的方法和技术（不局限于 Scrum）。

当你开始在你的组织中使用 Scrum 时，你也应该探索其他的方法，这会让你找到一些你可以使用的宝藏方法。

本书中定义的实践、原则以及所使用的工件，是能够被称为使用"敏捷"的最小单位。

也就是说，最少在初期阶段，最好是坚持一个方法。Scrum 就是一个非常适合初期阶段的方法，因为它简单易用。

之所以开始并坚持一种方法一段时间，是因为你首先需要集中精力让人事方面和组织向敏捷文化迈进。

过多的关注工件、工具或者技术，可能会转移人们对真正重要内容的注意力。敏捷首先是理念的改变，然后是组织和个人行为上的改变。

Scrum 中的一切都是为了促进敏捷实施的进展而特别设计的。别太快分心，分心被证明是具有破坏性的，而且将导致失败。建议坚持这些基本要素，直到组织的理念已经发生改变，对新工作方式的抵制已消失，然后再从其他框架中引入和尝试新的方法和技术。

要让敏捷在组织中发挥作用极度需要奉献精神和专注力。

在本书的附录中将探寻其他方法。目前，先让我们选择 Scrum 作为实施敏捷的重点方法。

2.1 敏捷实施过程中的挑战

与精益开发一样，对于敏捷而言，应该有这样的认识：是由整个组织创造价值，而不是单一的产品、服务流程，甚至也不是由单独的团队交付价值。整个组织必须将其所有跨职能流程组合在一起，以产品或服务的形式为客户创造价值。

在精益中，价值流必须映射到业务功能和流程中。不仅仅关注产品或产品待办事项列表（Product Backlog，下简称 PBL）是非常重要的，同时还需要放在更广泛的商业情境中。

同样，在敏捷开发中，除非团队能够创建一个多功能、多维度的解决方案，否则为客户实现他们想要的价值将成为不可能的挑战。

然而，成功的跨团队交付对组织的等级制度、组织结构和既得利益构成挑战。

Scrum 的发起者曾说过：Scrum 易理解，但是难以掌握。

困难的地方在于，要成功且高效地实施敏捷以及 Scrum，组织必须做出巨大的改变，而这种改变实施起来困难重重。推行敏捷或 Scrum 是一场不应被低估难度的大规模变革管理举措，它必须精心规划和细心执行。

关于变革的方法与技术多种多样，没有既定套路可言。如果发现某种方法/技术适合你，那么请继续使用就好。不过以往惨痛的经验教训告诉我们：渐进式的变革往往比革命性的变革更容易成功。

所以敏捷转型[7]的理念之一就是：耐心而不是急于求成。

变革管理不是本书的核心，因此主流变革管理方法将不再赘述。但是在首次在环境中引入敏捷时，提供两种概要描述的备选方法是非常有用的措施。

2.2 变革管理

变革是不可避免的，抵制变革也是没用的，拒绝采用以增量方式交付价值的敏捷方法也是同样徒劳无益的。

如果组织不处理变革中的人员问题，那么敏捷落地必将失败。

Jeffrey M. Hiatt 和 Tim J. Creasey 在他们的著作*《变革管理：变革中的人的一面》*（*Change Management：The People Side of Change* (2012)）中将变革管理定义为遵循可重复的循环、并使用一套工具集的过程和能力，因为它们能促使变革发生并提高组织的有效性。

他们定义了五项变革管理宗旨，帮助人们处理和接受变革。

1. 变革有因

我们通过变革实现一个特定的和符合预期结果的未来状态。变革的原因因组织而异，可能包括降低成本、提高客户满意度，以及在敏捷案例中谈到的更高的客户价值。

变革总是由面临的机会或需要解决的问题所驱动。

2. 组织变革需要个体的改变

仅仅是新工具或者是流程的引入并不足以实现组织的变革，这需要组织中的个体接受新的价值观，并且以新的方式工作。

3. 组织变革成果是个体变革结果的集合

变革是个体事件，因此需要考虑人为因素。这意味着员工接受度越高，组织就越能达到预期结果。

4. 变革管理是管理变革中人员方面的一个有利框架

抵制变革是组织的常态，尤其是在变革饱和的环境中。管理变革中的人员管理会推动更快的对变革的采纳速度和更好的变革熟练度。

5. 实施变革管以获取变革好处、实现预期结果

变革管理的首要目标，是推动和支持理想的未来状态和预期成果的实现。

---

[7] 译者注，这里作者用的是 agile mindset，但结合上下文，此处应该说的是 agile transformation，敏捷转型。

2.3 ADKAR®和 ADAPT

Hiatt 和 Creasey 发明了在个人层面帮助引导变革的 ADKAR 模型，是意识（Awareness),欲望（Desire),知识（Knowledge),能力（Ability)和强化（Reinforcement)几个英文单词的首字母缩写。当每一位员工都达成了这全部五个里程碑时，变革才算成功。

在本书中所描述的 ADKAR 或 ADAPT 模型或许是最接近于更传统的变革管理的方法。

其次则是精益管理原则和方法，稍后会有更多关于这个方法的内容部分。

ADAPT[8]以 ADKAR 模型[9]为基础，它是由 PROSCI 开发的流行变革管理方法，与原有方法仅有细微的改变。

ADKAR 是下面几个词的缩写：

- 意识（Awareness）
- 渴望（Desire）
- 知识（Knowledge）
- 能力（Ability）
- 强化（Reinforcement）

Scrum 大师 Mike Cohn 将 ADKAR 模型改进为 ADAPT，因为他认为在一个大多数工作都是以专业知识为主的环境中，知识和能力是密不可分的。所以，它把这两个词合二为一为能力。

Cohn 还认为强化（reinforcement）一词对于团队来说太模糊了，他使用两个词来替代——推广和转移。

因此，ADAPT 代表着：

- 意识（Awareness）
- 渴望（Desire）
- 能力（Ability）
- 推广（Promotion）
- 传递（Transfer）

下面是对于 ADKAR 的高层次描述：

意识（Awareness）

当你意识到现状已经不如人意，才能真正开始改变，但是意识到过去起作用的不再适用可能极其困难。

一般来说，人们通常对改变的必要性意识发展比较慢。这常常是由于缺乏对大局的了解。

---

[8] Cohn, M. (2010). *Succeeding with Agile: Software Development Using Scrum*. Addison-Wesley. Chapter 2.

[9] 更多关于 ADKAR 模型的信息可以在下方网站中查询：
https://www.prosci.com/methodology/adkar

实施 Scrum 的必要性可能是多方因素共同作用的结果，而这些因素并不为每个人所见。如果变革的需要仅仅针对少数人（比如，那些看到新客户销售额下降的人）明显可见时，关于变革的谣言就会开始传播。

组织中的每一个人都需要了解推动变革的真相是至关重要的，从而确保变革不会对组织产生消极行为和负面后果。

为了确保思维意识是基于事实而不是猜测，请确保相关的问题清楚地传达沟通，并提供基于客观衡量指标和事实的证据。

你需要向听众解释变革是如何解决问题的。通过运行一个试点项目来解决实际问题是一个好主意，这也可以证明变革确实能解决问题。

你的组织希望引入敏捷的原因可能有很多，但请清楚的意识到这个事实——专注于最重要的两、三个原因，而不是更多。

渴望（Desire）

如果意识到当前的流程不适合组织，或者不能为客户产生预期的结果，人们可能很难接受。

从项目管理的角度来说，我们大部分人要么学习过、要么有过传统项目管理方式的经验，这是毋庸置疑的事实。

既然传统的方式过去一直可行，那么何必要改变?

真实情况是，如果你花一点时间跟他人交谈，讨论一下他们在项目中失败的经历，你就会明显的发现，传统方式其实并不是一直起作用。

如果拿所经历的负面结果做比较，然后解释或展示敏捷是如何避免这些问题的，人们对于更积极的结果的渴望就产生了。

例如，如果你询问人们现在想要的跟两年前的是否一样，得到的回答会是否定的。对于瀑布式开发的项目也是如此，他们会意识到交付的产品或服务往往已过时，不再被需要，或者不再符合目前的需求。

然后，简单地将交付不再被需要的产品或服务的传统方法与敏捷方法进行比较。当使用敏捷方法时，整个团队将致力于当前需求，并在短周期内交付当前所需。这让他们可以说：是的，这就是我们想要的，比传统的方式要好。

为了引起变革的渴望，我们需要让相关干系人了解到，这里有一个更好的方法可以使用。与此同时，组织也迫切需要更好得结果。当你为转型敏捷提出一个很好理由时，干系人也能跟你达成一致。

然而，请注意：不要表现过去的好像都是错的一样；应该关注于真正那些能解决迫在眉睫问题或者立竿见影效果新优势的方面即可。还有一个更好对的做法是，以一个试点项目展示它是有效的且运行良好。请务必要做到这点!

随着事情开始改变，请帮助人们解决向新的方法转变过程中遇到的困难。要有耐心并帮助团队战胜困难。要让干系人明白，变革并不可怕，而且非常安全。

尽可能地邀请干系人参与进来，有时也可以考虑针对变革的激励措施。然而，将激励作为变革的驱动力需要依赖于变革发生的背景和文化。对这一点务必要保持敏感。

Tom Peters 曾有一句名言：世界上最困难的事不是让人们接受新事物，而是让他们忘掉旧的。

让人们忘掉旧事物的唯一办法就是——给他们展示更好的来替代掉它。

如果不具备敏捷的能力，那么这所有的意识和渴望都没有任何用处。

能力（Ability）

就如之前 Tom Peters 的例子一样，要想成功的实施 Scrum，不仅需要团队成员学习新的技能，而且要让他们忘掉之前的做法。

- 下面是一些 Scrum 团队将遇到的大挑战：
- 学习新的技术技能
- 学习以团队的方式思考和工作
- 自我管理的概念
- 学习如何在一到四周的较短的、固定周期（时间盒）创建可工作的产品或服务

应该对提供适当的培训给与重视。然而，现实是培训是不够的。在初期，团队是需要被手把手地指导和训练的。为此，建议从公司外部聘请一位经验丰富、并且有过成功实施 Scrum 的人来担当团队的 Scrum 教练。

不要期待太多，也不要期望进展太快，尽管你希望人们能承担他们的职责，并且以新的、正确的方式来完成，但是不要设定不合理的目标。

检视、度量、提供反馈，并邀请干系人参与问题的解决。

请记住，任何训练都不足够让你和你的团队做好准备；你必须从头开始就把从工作中学习、在犯错中学习，作为学习的一部分。如果 Scrum 团队是初次接触 Scrum，最好是严格遵循规则，直到有足够多 Scrum 和敏捷工作经验。

推广（Promotion）

你知道如果你不对你的观点进行推广，你的公司就不会接受。变革又怎么会例外呢？

我们必须持续不断的推广新方法或者放弃旧有方法的好处，以此来确保 Scrum 这种新方法，可以在组织中恰当地融入和制度化。

推广阶段有三个目标：

1. 你很可能以小的、试点的项目来为下一步地推广 Scrum 奠定基础。促进当前的成就和提高对新方法的意识，这将启动下一轮的改进。
2. 第二个目标是通过传播这些试点团队所取得的好处来增强现有团队中的敏捷行为。
3. 第三个目标是在参与和使用 Scrum 的团队之外的人中建立对 Scrum 的意识和兴趣。

推广意味着宣传成功案例、营造吸引他人渴望加入的氛围。有时，邀请观众参与互动展示，也是促进变革的好方法。

让干系人参与游戏和模拟之类的活动，能让他们获得第一手的经验，而且不用经受现实中产生任何痛苦和负面影响。

传递（Transfer）

如果你已经做了试点，并且取得了巨大的成功，你需要在组织范围内传播敏捷。如果组织中的其他人没有跟进的话，那么你将无法保持势头。为什么会这样？因为敏捷涉及除了不同的做事方法外，还包含了不同的思维方式。如果旧的思维方式仍然残

留在组织中的其他部分的话，它可能最终会扼杀掉组织已取得的、哪怕是最好的开端。

如果不把使用 Scrum 的影响转递给其他部门，组织免疫系统[10]将不断攻击变革，并且最终导致变革的失败。

这并不意味着组织的其他部门也需要开始使用 Scrum，只是说它们至少要跟 Scrum 兼容。

在这种情况下，不断变化的指标、人力资源和其他政策、甚至是治理和控制结构，都不应该被忽略。你不能以度量传统组织结构的方式来度量敏捷团队的成功。你必须使用新的管理方法、度量方法和团队组织方法来替代旧的方法。管控措施需要改变，这其中最根本的是项目与项目集管理，人力资源政策以及角色和职责、管理架构、绩效管理，产品管理，甚至是组织中的资金、预算和财务实践。

Scrum 的引入将影响整个组织，最终组织中的一切都将改变，并且越早改变效果越好。

关于 ADAPT 的结论

与 Scrum 本身一样，ADAPT 也是迭代渐进的方法。它由"变革是必需的，现在的工作方式不再能产出可接受的结果"意识开始。随着这种意识的传播，一些早期拥护者，将会在试图改进解决方案的时候发展出尝试 Scrum 的渴望。

通过反复试错，组织中的早期拥护者将具备采用 Scrum 来获得成功的能力。这时候，新的状况可能会出现：在没有完全使用 Scrum 的大型组织中存在着少量成功实施了 Scrum 的团队。

随着这些早期的团队不断改进使用 Scrum 的方法，他们开始推广他们成功的方法——有时在非正式的场合，比如跟另外一个团队的朋友吃饭的时候；有时会在非常正式的场合，比如部门级别演讲或者计划中的变革项目。

这些做法有助于其他团队中的个人开始从意识（Awareness）到渴望（Desire），再到能力（Ability）的阶段地发展，然后，很快地这些团队也将开始推广（Promote）他们成功案例。

2.4 什么是产品

在 Scrum 中，"产品"是一个兼容性的术语。有些人是这么定义产品一词的：产品既可以是商品也可以是服务；事实上，在许多情况下，产品既是商品，也是服务。

还有人将服务与产品区分开来，常谈论产品以及服务；在这种情况下"产品"不再是一个兼容性的术语。

本书基于公认的 Scrum 实践，因此对"产品"一词的立场是：在 Scrum 中，产品是一个兼容性术语，因此，服务也是产品。

[10] 可以理解为组织旧有工作模式于思维方式——译者注

3 精益管理

ADAPT 可以被视为一种变革管理方法。另一种促进变革的形式则更具组织性和演进性。

将精益作为一种管理和运行一个组织的方法，意味着管理风格、员工行为、组织结构以及角色和责任定义方式的彻底改变。

然而，精益是一种渐进的管理方法，它并不试图促进革命性的组织变革。这意味着通过精益进行企业变革需要花费大量时间，因此需要组织内领导层坚定不移的承诺。

所有敏捷方法都是建立在精益所奠定的基础之上，构建组织敏捷性和成为精益组织在其核心上是相同的。

本文的目的不是传授精益，而是分享一些核心的精益原则和实践，这些原则和实践构成了成功实现敏捷的基础。

3.1 精益，一种战略和管理的方法

大多数情况下，精益可以被理解为一种运营战略以及一种设计、转型和运营企业的方式。精益聚焦于两个主题：

- 客户价值最大化
- 减少浪费

二战后，日本面临严重的资源短缺，丰田的领导层意识到，一系列创新是提供流程连续性和产品供给多样性的唯一途径。

丰田专注于最大限度的减少生产汽车所需的原材料，同时缩短客户订单和车辆交付之间的时间。

所采用的理念与方法被称为丰田生产体系（TPS）。该系统将制造工程师的关注点从单个机器及其使用转移到基于客户需求的整个流程中的产品流。

丰田很快发现，工厂员工的贡献远不止体力劳动；运行和操作机器的人最有可能提出创新的想法并改进制造流程。

两个基本原则构成精益的基础：

- **及时生产**：只在需要的时候，生产所需的产品
- **自働化（Jidoka）**：规范化和自动化

随着丰田的改进，TPS 成为 2001 年出版的包罗万象的 The Toyota Way 的一部分。如今，自働化（Jidoka）和及时生产原则仍然保留。然而，他们（自働化、及时生产）现在所属的丰田模式理念包含了两个更高原则：

- 尊重员工
- 持续改进

精益为当今行业提供了根本性改进：从设计、初始设定到运营和转型。这些概念围绕着为客户和干系人建立长期的结果、企业范围的过程能力的建设与调整，维持成功的文化因素，持续改进涉及到每一个人、每一天、每一个地方。

精益企业比非精益企业更成功。他们创建了更高的商业价值，更容易留住人才。他们还发现，随着时间的推移，维持成功会变得更加容易。

图 2：精益原则

图片由 EXIN 基于以下内容创建：Botha, J. (2021). *Lean fundamentals for IT [Courseware].* GetITright.

首先，精益有助于聚焦客户价值，通过这样做，企业可以为其产品和服务增加更多价值，同时减少浪费的来源、提高他们的敏捷性和适应性能力。通过与客户和终端用户建立更好的联系和改善对话，帮助企业提高客户的满意度，并大幅提升客户的忠诚度。

其次，精益组织不断提升过程绩效。他们的服务质量更高、交付时间更短、项目和运行效率不断提升。

精益组织还意识到，如今大多数员工都是知识型员工，一个组织最重要的资产是它的员工。在精益组织中，员工的参与程度更高，员工的积极性和工作满意度也更高。

最后，**减少过程浪费、优化增值工作（腾出时间增加更多价值）**可以带来财务价值。如果组织不了解整个组织用来创造价值的"过程"，那么他们就不可能做到这一点。一种叫做**价值流图（VSM）**的技术提供了必要的洞察力。通过该技术我们不仅可以对价值进行改进，还可以消除其中的浪费。

减少浪费的一种方法是缩短订单接收和交付之间的持续时间，通过精简系统中的工作流程，并将所有流程和系统改为**拉动**式系统而不是**推动式**系统。

改进工作流还可以改善现金流，因为它降低了对原材料库存的要求。必须强调的是，提高盈利能力并不是精益的首要目标。尽管其可以预期，但它是在改进和减少花费在非增值活动上的精力和时间后的一种附加产物。

低质量也可以是采用精益的推动因素。低质量对组织内外都有影响，如名誉受损和客户信任的丧失等方面可能会导致组织支付巨大的成本或遭受处罚。此外，人才的浪费或意料之外的缺陷数量也会在组织内部造成压力。同样，由于检查、返工或消极怠工，成本也会增加。

精益改变了范式。下面让我们以"知识就是力量"这个传统的范式为例。过去一个企业可能发现自身依赖于一些知道流程如何运作的关键资源。而在精益的范式中，与其依赖少数人，不如确保所有员工都是熟练和有能力的。

所有敏捷和 Scrum 的原则和工作方式都是基于精益原则和实践，包括自管理团队的概念、决策权的下放、创造让每个人都能做出贡献并说出自己的想法的安全环境，以及激励所有人通过任意的方式学习，包括从失败中学习，而没有惩罚性的结果。

4 Scrum 与持续改进

敏捷和 Scrum 本质上是关于持续改进的；这就是为什么敏捷方法是迭代的。

像戴明循环（PDCA）这类持续改进方法的核心是经验主义的思想和持续不断学习的概念，以及在学习过程中产生的改进。这些核心也是精益和敏捷的核心主题之一。

敏捷可以被视为实现持续改进的一种方式，而不是执行项目的一种方式。

来自企业，客户或用户的新需求推动了一个永无止境的持续改进计划，同时也为干系人创造了更好的价值。

改进是为现在重要的事情和解决眼前问题的灵活性所驱动 - 如果需要，甚至可以改变当前产品的方向！

但是，这应是以有计划和可预测的方式进行。它与如何在组织中创造与维持一种变化和价值交付的节奏，并将其保持紧密相关。

Scrum 也在使用它的环境中创造了可预测性和稳定性，因此我们可以使用 Scrum 作为一种持续改进的方法来维持积极变化的节奏。

5 Scrum 基础知识

5.1 Scrum 简介

作为一个高阶框架，Scrum 创建初衷是为了帮助组织以增量的方式交付复杂项目。作者之所以特意把 Scrum 定义为一个高阶框架，其目的是为了让组织发展出符合自身情况的研发方法来填补 Scrum 中未谈到的一些细节空白[11]，而这种研发方法一定是在当前组织中最有效的。

相比较于 SAFe、Disciplined Agile Delivery（DAD）、ABC/DSDM 而言，很明显的看出 Scrum 是一个轻量级的框架，这非但没有降低 Scrum 的实用性，反而还有所提升。从这里你一定可以看出 Scrum 是一个非常强大的工具，它让你可以自由的发展出适合你的敏捷方法。

作为一个轻量级的高阶框架，Scrum 没有许多强制性、详尽的规范，但是其精心设计的基础理念经过多年来的反复验证，已经成为今天 Scrum 可以成功实施的坚实基础。使用 Scrum 的最好方式是保持其基本原则的同时因地制宜。

在使用 Scrum 框架的最初几个月里，最好的方式是根据其指导原则行动，而不是急于修改它。有时候我们很难能在一开始就理解为何 Scrum 建议某种做法，只有在经过一段实践的尝试验证使用之后，才能够认识到到使用 Scrum 推荐的方法是明智之选。

Scrum 常常被定义为：

> 一个能够帮助团队在面对复杂适应性问题时，可以有效地、创造性地交付高价值的产品的框架。Scrum 的特点：轻量级、易理解，但是难以掌握。

最后的"难以掌握"看起来比较有趣，这似乎有个悖论：为何 Scrum 高阶且简单直接，但又难以掌握呢?

其原因多种多样，但是以下几点是重中之重：

1. 它与传统管理方式不同。
2. 它与传统组织架构不同。
3. 它信任以及依赖团队成员的表现，这在传统管理组织中并不常见。
4. 它并没有统一的执行标准，因地制宜至关重要。

简而言之，在使用 Scrum 时团队需要做出巨大的改变，这在初期的时候可能困难重重。但是经历一段时间后，组织及其成员拥抱了这种变化，且感受到了它为组织、客户和内部员工带来的收益，就很难再回到传统模式了。

在使用 Scrum 时，第一个重大组织变革就是，所有的工作都由一个自组织[12]的团队完成，这相对传统的组织管理架构而言可能是一个非常陌生的理念。

---

[11] 比如站会中要沟通的信息——译者注

[12] 在 Scrum Guide 2020 版中，出现了新的自管理（self-managing）一词来描述团队。——译者注

在 Scrum 中只有三种角色，并不存在传统的管理者角色。

第二个重大改变就是，在项目开始之前不会做详尽的项目规划，也没有截止日期或者甘特图。这点曾经备受质疑。

起初，管理层可能对此持怀疑态度，因为这似乎是一个没有人愿意承诺的空白页。

传统管理方法的问题是：即使你做出了承诺，并且为此制定了详尽的计划，但在实际执行过程中，现实与计划不一致是家常便饭，因此承诺也常常无法遵守。我们都知道这就是现实情况，但是我们只是必须承认这一点。

一个项目计划无法做到完美，原因往往不是因为项目参与者不够努力。相反，如果假设在一个复杂自适应系统（complex adaptive system，CAS）——比如某个组织内——可以对未来的变化进行完美的规划，这个假设本身就是有问题的。

Scrum 承认提前制定一个完美的计划是不可能的，因此它提供了一个更优秀的解决方案：在更宽泛的（轻量级）计划中持续致力于对业务最重要的事情。这使我们能够专注于业务价值实现，而不是时间节点。

5.2 独树一帜的工作方式

在 Scrum 框架及干系人法中，使用只有三个简单的职责（角色）的自组织团队，少量的规划事件和工具（工件），以及一些简洁明了的规范。

这些组件是 Scrum 的基本元素，也是 Scrum 运转的必要条件。虽然在使用 Scrum 时，组织可以进行自由调整和改变这些元素。但经验表明，如果改变了这些基本要素，很容易导致全盘崩溃。

建议遵守 Scrum 框架中基本要素和相关的"规范"。每个要素都有其特定目的，且对框架的成功均至关重要。

Scrum 是为了管理软件项目而生，但是对于许多组织而言，在不同情境下的项目使用也同样有效。

如果迭代式价值交付和增量式的知识转移是当前奉行的规范，那么 Scrum 就是实现这种规范的方法。

在 Scrum 的术语表中，任务执行者组成的团队被称为软件开发人员。而 Developers 一词又是 Scrum 中三个职责之一，但这并不意味着团队仅仅由开发人员组成。因此，在这种大背景下，应该将其解释为能够开发、构建、交付特定冲刺或迭代目标的团队。Scrum 团队包括 Scrum Master, 产品负责人和所有的 Developers。

无论你的组织是软件企业、制造业、服务商甚至是学校，学习都应该是一件需要反复进行并且持续渐进的事——因此 Scrum 被应用于与时俱进的学校，并作为主要教学方法。请记住：Scrum 是一个始终能为小型、自组织、跨职能团队创造价值的方法。在这种大背景下，这些团队并没有因为成为了大团队而失去了自主性。相反，她们在各自称为自主团队的同时，还进行了关系网络、协作以及互通有无的方式进行合作。因此，Scrum 也能够对采用该框架的传统等级制度企业产生重大影响。

5.3 Scrum 背后的基本原理

请注意，这本书是为已具有 Scrum 基础知识的人士的进阶学习而编写的。如尚未掌握基础知识，请阅读 *Agile Scrum Handbook*（Rad, 2021）.

此书的目的是为了帮助您成为一个更优秀的 Scrum Master （以下简称 SM）或 Product Owner（以下简称 PO）。因此本书将重点关注 SM 和 PO 的职责及所需实践。

请注意：SM 应该了解产品负责人，反之亦然。

Scrum 背后的理念并不新鲜，可以追溯到经验过程控制、精益中的原则及实践。

我们的观点是：根据已知经验而不是根据猜测来做决策以及执行，并以此来培养知识和经验。这也就意味着当我们迷惑不解时，经验主义可能是唯一的解决方式。

在科学领域，经验主义的基础是：提出一个假设，并创建一个实验测试它。经过测试之后，假设还要么被证明，要么被推翻。无论如何，你总有所得。

这也是 Scrum 相比瀑布方法优势显著的地方之一。在瀑布方法中，所有的项目规划都被提前完成，而这大多基于往往被证明是错误的假设之上。

我们从来不否认"在 Scrum 中我们有所不知"的事实，我们勇敢承认，并使用经验主义尽快找到解决方案。

这就是经验过程控制三大支柱之一：**透明**。我们在试图了解需要做什么的时候学到的经验教训是另外两大支柱：**检视**和**适应**。

5.4 Scrum 三大支柱

也许是时候重新审视 Scrum 三大支柱了：

- 透明（Transparency）
- 检视（Inspection）
- 适应（Adaptation）

透明（Transparency）

透明意味着工作流程和将要完成的工作必须对团队成员可见，因为他们将对完成的工作及其结果负责。

如果使用通用标准、语言、方法、工具、度量指标和通用词汇，每个人都更容易理解。工作内容也必须清晰可见，以便每个人都能清晰看到和明确理解需要被完成的工作。

检视（Inspection）

检视意味着我们必须持续检查正在做的工作是否有效、所做的假设是否真实有效。一旦工作完成，必须经常检视、验证其工作方式和相关的 Scrum 制品，是否需要更改、修复或是改进。检视是三大支柱中的最后一步。

适应（Adaptation）

适应意味着一旦过程中的任何元素，或者工作本身偏离了决定的规范，就必须对过程或者工作本身进行调整，以确保迅速恢复到既定的规范和交付的质量。

5.4.1 Scrum 事件

Scrum 只有五个活动或仪式[13]，每个活动都提供了检视和适应的机会。

这五个活动分别是：

- 冲刺
- 冲刺计划（sprint planning）
- 每日站会（daily scrum）
- 冲刺评审（sprint review）
- 冲刺回顾（sprint retrospective）

值得注意的是，尽管产品待办事项列表（product backlog）梳理会没有被定义在其中，但它仍旧是一项非常重要的事件。

文化变革对于组织文化有着显著的影响，许多组织在采用 Scrum 或者其他敏捷方法或框架中往往会低估它。

众所周知，组织文化变革不是一朝一夕的事，但是要想 Scrum 和敏捷发挥其作用，组织必须接受它们的价值观。

可能正是在这一点上，大多数放弃了变得更加敏捷以及使用敏捷方法的目标的组织都失败了。

请牢记：Scrum 的核心是经验主义及精益思想。它基于“知识来源于经验，决策应基于观察到的内容和所学知识”的理念。精益思想帮助我们聚焦于有效的方法并保持简单。精益中减少浪费的方法，不仅仅可以用于资源消耗，还可以用于构建新的、有价值的产品及服务的过程。

Scrum、精益以及其他敏捷方法都依赖于一组价值观才能有效的工作，因此明智的做法是问自己：这些价值观是否与组织价值观或文化相容？这些价值观偏离得越多，转型敏捷或者采用 Scrum 的困难程度就越高。

但是这些价值观到底是什么呢？

5.5 Scrum 价值观

Scrum 团队作为一个自管理、自治的单位运作，为正在进行中的工作负责。只要他们还在为客户交付价值，即使团队出现了交付失败或者其他错误，为了他们的学习和成长，也应该让团队从错误中进行学习。反过来说，交付价值意味着团队能够诠释和确认客户需求，并确保通过将需求拆解成可被执行的任务的方式来实现价值交付和相关成果。

这里的关键问题在于：团队应该感受到安全感，并知道只要他们从错误中吸取教训，就不会受到惩罚。追责可能是某些组织采用 Scrum 或敏捷失败的最主要原因。

[13] 一般现在流行的用语是时间，“仪式”这个词使用频率较低。——译者注

Scrum 的五大价值观分别是：

- 承诺（commitment）
- 勇气（courage）
- 专注（focus）
- 开放（openness）
- 尊重（respect）

如果这些价值观得不到理解和实现，那么三大支柱也将崩溃。

此外，团队成员以及与他们打交道的每一个人，包括组织内部其他成员以及他们的客户，也应该相信和支持这些价值观。

本质上讲，不仅仅是团队成员接受和相信这些价值观，其组织也应该保持一致。

让我们看看这些价值观在实践中意味着什么。

承诺（commitment）
承诺意味着团队中的每一个成员都致力于团队及其承担的工作。

勇气（courage）
团队成员必须表现直言不讳的勇气，而其他成员则必须用心聆听。团队必须尽快解决问题以便兑现承诺。

专注（focus）
团队成员无论是单独工作的还是集体共同工作的都能专注于完成工作，以便在限定范围内交付所承诺的内容。

开放（openness）
我们每一个人都会犯错，而且也没有一个人是万能的。团队成员无论是犯了错，还是遇到阻碍，或者不知道如何执行分配给他们的任务时，都应该敞开心扉并承认。隐藏错误将会引起一个团队无法允许的连锁效应。要预防这种情况发生，必须让团队成员相信不会因为毫无保留地开口说话而受到惩罚。不幸的是，这种情况[14]在许多组织中仍旧是常态。

尊重（respect）
团队成员必须相互尊重，并随时随地的帮助团队其他成员。

5.6 Scrum 职责摘要

5.6.1 Scrum 团队

Scrum 的基本单位是一个被称为 Scrum 团队的单位。它由一名 Scrum Master（以下简称 SM）、一名 Product Owner（产品负责人，以下简称 PO）和若干 Developers（开发人员）组成。在 Scrum 团队中，没有子团队，也没有层级之分。它是一个具有凝聚力的、专注于目标，即产品目标的专业团队。

[14] 指毫无保留开口说话而受到惩罚。——译者注

Scrum 团队是一个自管理、跨职能团队，他们能在不受外部影响的情况下，选择他们一致认同的、最好的方式达成目标。在团队中，PO 代表客户与业务，团队成员通过 PO 来了解业务的优先级。

Scrum 团队应该由跨职能的 Developers 组成，团队成员应该具备完成冲刺（冲刺是一个时间盒事件，即具有固定时长的事件）中所选择工作需要的所有技能，团队对团队外部技能的依赖程度应相当低。这也就意味着团队尽可能少地使用外部 Developers，只有当缺乏某些特定的必要技能时，才可引入外部 Developers。这还意味着在小型团队中更加容易实现 Scrum 价值观。与小团队相比，大型团队在实现 Scrum 价值观的过程中，往往会产生很多缺陷。

Scrum 团队应该足够小以保持 agile，又应该足够大以完成冲刺中的大量工作。建议单个 Scrum 团队的人数保持在 10 人以内。

为什么 Scrum 团队规模如此之小？事实证明小的团队更容易沟通，有害的、过度的技能专业化不太容易发生，而且工作效率更高。

如果团队太大，应该考虑将大型团队重组为多个有凝聚力的 Scrum 团队，这些团队都聚焦于相同的产品。由于团队聚焦于相同的产品，因此它们应该共享相同的产品目标、产品待办事项列表（Product Backlog，下简称 PBL）和 PO。

Scrum 团队负责所有产品相关的活动，这当中包括干系人协作、需求核实、维护团队运作、维护团队进行的实践操作，同时不断尝试持续改进这些实践。为实现这一目标，团队需要投入时间和精力到调研、改善及其他任何有可能帮助改善其工作方式的事情上。Scrum 团队被授权管理自己的工作（自管理）。与此同时，团队也应该广泛征求意见，以便了解应该做什么以及什么才是真正对客户和用户有价值的事。这项协议[15]是定义 Definition of Done（完成的定义,后续简称 DoD）的基础。DoD 是 Scrum 团队绩效的关键指标。

整个 Scrum 团队负责在每一个冲刺中创建一个有价值的、有用的增量（increment）。

一个冲刺是一个时间跨度 1 到 4 周的事件。在这期间，Scrum 团队将开发一到多个价值增量。

一个有用的增量意味着能为干系人创造价值的、可以部署和使用的东西。

请注意：一个冲刺可能产生多个有用的价值增量。

Scrum 定义了 Scrum 团队中的三个具体角色：

- Developers
- 产品负责人
- Scrum Master

稍后，本书将具体阐述什么是有效团队、团队组成以及如何组建团队的更多细节。确保 Scrum 团队以可持续发展的持续性地工作的主要方法是使用冲刺。因此，冲刺是 Scrum 的基本要素。

[15] 这里说的是 Scrum 团队 负责所有产品相关的活动。——译者注

以下是*一个好*的 Scrum 团队应具备的基本要点：

- **Scrum 团队是自管理的**，没有人告诉 Scrum 团队应该如何将产品待办事项列表条目（Product Backlog Items，下简称 PBI），也就是需求，转化为创造价值的增量（产品/服务）。
- **Scrum 团队是跨职能的**，团队成员拥有团队创建产品增量所需的所有技能。这是理想状态，并不总是切实可行。
- **Scrum 不设置任何头衔或职位**，无论该团队成员正在执行何种工作。SM、PO 以及 Developers 不像传统的"角色"那么严格，而是一种职责。在某个时间扮演某种"角色"仅仅意味着某人需要确保某些事情完成。
- **Scrum 不设置任何子团队**，无论其所属领域为测试、架构、运营、商业分析，或是其他。
- 个别 Scrum 团队成员可能具有专业技能和关注领域，**但职责属于 Scrum 团队全体**。SM 和 PO 不能由同一个人担当，职责分离对于避免责任冲突非常重要。
- Scrum Master 和产品负责人的职责不能集中在同一个人身上，在这种情况下，职责分离对于避免责任冲突很重要。

5.6.2 Developers

Developers 致力于在每个冲刺阶段中创建可用增量的各个组成部分，并拥有开发增量所需的特定技能。

Developers 的技能、能力甚至技能水平往常差别很大，这主要由冲刺中涉及的工作领域决定。

Developers 的主要职责：

- 创建冲刺计划和冲刺待办事项列表（Sprint Backlog，下简称 SBL）
- 遵循 DoD（definition of done, 完成的定义）来提高质量
- 必要时调整计划以及工作方式，以确保冲刺目标（Sprint Goal）的进展
- 作为专业人员共同为工作负责

5.6.3 产品负责人

产品负责人（Product Owner，以下简称 PO）是负责通过与 Developers 和 Scrum Master（以下简称 SM）合作实现产品价值最大化的个人，而不是委员会[16]。

PO 最好是全职工作，这是为了避免多任务并行处理，以至于 PO 成为项目中的潜在瓶颈。PO 代表了产品待办事项列表（product backlog，以下简称 PBL）中所有干系人的需要（needs），所以他们必须掌握如何履行产品所有权相关职责的能力。如果想对 PBL 进行调整，你可以根据改变的价值说服 PO。

- PO 负责创建、管理和维护 PBL，以及管理产品预算和产品发布协调等工作。
- **PO 开发并明确传达产品目标，明确定义由需求组成的 PBL**，以此来确保产品目标的实现。需求被定义成待办项条目（Product Backlog Item，下简称 PBI）。
- PO 还需要根据组织目标来对 PBI 进行排序。

[16] 即不能是一个组织。——译者注

    - 请注意：排序需要由业务的优先级来决定，但它不是唯一可用的指标。
- **PO 作为客户之声（voice of the customer，简称 VoC）**参与 Scrum 会议、解释需求，并帮助 Developers 更好的理解 PBI 以及其优先级顺序。
- **PO 通常还需要在不同的 Scrum 团队之间进行协调**，以确保工作不重复、依赖关系对所有人清晰可见，所有的工作和关系依赖是一致且精心规划的。
- **PO 需要确保 PBL 经常性得到梳理。**PBL 梳理包括：重新排序、将需求拆分得详略得当，并且在 Developers 提供反馈和帮助下完成梳理工作。

由于这种职责的重要性，产品所有权不得分配给没有或者几乎没有得到客户授权和尊重的人。

PO 自行履行职责，但是也可以将责任委托给团队中其他成员。

虽然没有明确说明，但建议将业务中的产品所有权（战略、战术和预算）合并为由同一个人负责。

PO 不仅仅是业务代表，还是客户代表。这意味着他们应该来自业务/客户，而不是项目或技术。Scrum 指南（Scrum.org, 2020）特别指出，要使 PO 取得成功，整个组织必须尊重他们的决策。

PO 的决策应该在 PBL 内容及排序中清晰可见，并且以增量的方式在评审会中被检视和评审。

PO 的决策就是业务决策。拥有这种权威和影响力的 PO 被证明是最有效的。

5.6.3.1 PO 作为客户之声（Voice of the customer 简称 VoC)

客户之声（Voice of the customer 简称 VoC）是精益中用于描述客户期望、偏好和需要（needs）的术语。

作为客户的代表，产品负责人应对客户和用户的愿望和需求有深入的了解。因此可以说，PO 代表了客户和用户的声音。

但是，Developers 仍然需要在迭代期间与用户和客户接触，以便更好的理解和梳理需求与需要。PO 绝不应该成为阻碍端到端沟通的看门人。

作为客户代表，Scrum 团队应该遵循以下几项**敏捷原则**：

1. 我们最重要的目标，是通过持续不断地及早交付有价值的软件使客户满意。
    a. 请注意：在这个语境下，它可能指代任何产品和服务。
2. 欣然面对需求变化，即使在开发后期也一样。为了客户的竞争优势，敏捷过程掌控变化。
3. 业务人员和 Developers 必须相互合作，项目中的每一天都不例外。
4. 以简洁为本，它是极力减少不必要工作量的艺术。

5.6.3.2 优秀 PO 的品质

优秀的 PO 拥有相同的品质。以下是一些关于如何成为一个高效的 PO 的指导意见。

1. PO 是**强大的沟通者**，能够与每一个受众进行良好的沟通。他们调整自己的语言和沟通方式，从而以最适合的方式与受众进行沟通。更重要的是，优秀的

PO 都是很好的倾听者；他们倾听、观察、提出探究性的问题并反思他们所学的东西。

2. PO 应该**对业务有深入的了解**，以便他们能够有效地将业务需要转化为行动。他们来自于业务方而不是项目方，拥有正确的洞察力，并且能够在干系人需求不明确、甚至更糟的相互冲突的情况下做出复杂的决策和妥协。
3. PO 必须有足够的资历。只有这样，当他们代表企业做决定时，**能得到企业的尊重**。
4. PO **还需要有能力区分"客户需要的"和"客户想要的"**：如果有可能，干系人会向 PO 提出很多他们的需求。PO 必须能够区分什么是需要的，什么是值得拥有的。选择使用 MoSCoW 技术[17]对 PO 而言往往是一个好的开始。
5. **解决方案思维**：业务的许多需求或需要都以负面形式（问题）来表达。PO 必须有能力促进讨论，而讨论的重点不是问题本身，而是如何解决问题。
6. **结果驱动**：产出并不重要，必须要实现的是成果（也就是说，用户、组织、客户在交付增量之后能做什么、获得了怎样的价值）。

其他的一些技能与特质也能够帮助成为一个高效的 PO，但是上述几项是必不可少的。

5.6.4 Scrum Master

Scrum Master 的职责与传统的项目经理大不相同。

Scrum Master 与橄榄球运动中的前锋（Scrum-half）很像，而 Scrum 一词也正是来源于此。他们的工作是确保在列阵争球（Scrum）的过程中得到球，并履行自己的职责，将球传给后卫（为业务创造价值），然后后卫需要用球来得分（达成目标）。幸运的是，如果你不熟悉橄榄球比赛，那也没关系，你不需要先学习它的规则就可以掌握 Scrum 框架，所以不用过分强调这个类比。

Scrum Master（下简称 SM）帮助团队成员协调工作、组织各项会议、帮助团队克服困难，以及使用可视化工具跟踪项目进度，以便使团队中的每一个人都能确切地了解到底取得了什么进展、仍然存在哪些问题、如何在团队成员有需要的时候帮助他人。

Scrum Master 为帮助每个人了解 Scrum 理论和实践来促进、支持 Scrum 方法和流程负责。我们可以说 SM 有三方面的作用：

- 培训者
- 引导者
- 教练

担任教练和引导者的角色时，SM 不仅仅应该只注意 Scrum 的技术方面因素，人的因素也非常重要，尤其是在初期阶段。他们应该作为一个变革派，创建一个有凝聚力的团队，并且应对具有破坏性的团队动作。

SM 是服务于 Scrum 团队的领导者（服务型领导者）。他们是通过：

- 确保目标、范围和产品/服务得到理解
- 通过寻找改善产品待办事项列表（简称 PBL）
    - SM 并不需要管理 PBL，而是要提出建议从而让 PBL 被更好的管理。

[17] 关于 MoSCoW 的详细说明,请查阅 MoSCoW 部分。

- SM 帮助 Developers **理解 PBI** 内容、顺序、优先级以及其价值。
  - 虽然这主要是产品负责人（下简称 PO）的职责，但也需要 SM 的协助。
- SM 还需要帮助团队成员理解 **Scrum 以及其所有组成部分**，并帮助他们在团队和组织环境下以最好的方式使用和适应 Scrum。
- Scrum Master 是 **Scrum 事件的引导者。**
- 经常被忽视的是，**SM 需要保护团队不受外界干扰和分心，消除阻碍（obstacles）和障碍（impediments）**以提高/支持团队的表现，同时还要促进沟通。
  - 在这个角色中，SM 还需要在确保其工作内容的前提下，建立一个最适合团队工作的环境。

SM 最好不要同时兼任 Developer 的角色。这种做法会导致其缺乏对 SM 自身职责的关注，并且可能导致其他的障碍。

在其他敏捷方法中，SM 被称为项目经理（这不是个好主意）、迭代经理、敏捷教练或团队教练。

担任教练角色时，SM 凭借他们在 Scrum 方面的经验，帮助团队找到适应特定背景和情况下实施敏捷原则和 Scrum 方法、技术的最佳方法[18]。

使用没有经过适当培训的现有项目经理来担任这 SM 一角色时，一定要特别小心。如果你采用通常在瀑布项目中使用的典型命令与控制架构时，你将会迎来失败，而且是非常惨烈的失败。

另外请注意：使用前技术领导作为 SM 也可能产品负面影响。因为在作为团队领导的角色中，他们习惯性的根据以往经验提供关于"必须做什么"的结构化指示；但 SM 并不代表 Developers 做出决定。在考虑由谁来担任 SM 时，建议优先考虑受过相关职责培训，并且自愿担当的人。特别是在实施 Scrum 初期时，最好引入一个称职的、拥有丰富经验的教练来推进组织转型的进展。Scrum 不仅在实践方面是全新的，在工作方式方面也与大多数组织的传统工作方式大相径庭。

在转型过程中，你将面临很多障碍。有经验的教练将告诉你什么是有效的、什么是无效的、学到的经验教训，以及不惜一切代价都要避免的事情。

经验丰富的教练首先是组织变革管理大师，然后才是 Scrum 专家。专家的身份是必须的——能够帮助组织渡过艰难转型期的超级专家。

5.6.4.1 优秀 Scrum Master 的品质

前面我们已经谈到了 SM 的职责，以及成为一名优秀的 SM 应该具备的隐藏技能。

以下是高效的 SM 所应该具备的品质和要求。其中某些品质相对于其他来说会偏"软"。但让我们先从 Scrum Master 所应具备的硬技能开始。

[18] 敏捷中说最好是个比较敏感的事情。——译者注

SM 是教练和老师，这就暗示着他们必须有扎实的 Scrum 知识和敏捷理论知识。

- **SM 是终生学习者**，并不断的扩展他们的技术和理论体系。这不仅仅包括 Scrum 和敏捷的理论和实践，还包括其他团队用来创造良好效果的最新工具和技术，并且展示组织或其团队何时、以何种方法可以从中受益。
- **SM 是组织者，不是监督者**；他们需要以协商的方式来构建和管理团队用来完成迭代的系统。这意味着 SM 应该是一个处事井井有条的人，当他们周围的一切都出现问题时也能保存冷静。
- **SM 他们必须深入了解团队使用的工具**，因为他们通常会负责工具和数据的维护，比如 Scrum 看板和燃尽图。
- **Scrum Masters 必须是一名优秀的讲师或者教练。** SM 应该知道要做什么，还应该能向每个参与者解释在 Scrum 中如何以及如何使用特定的方式来做事情。
- **SM 最初需要确保团队掌握 Scrum**，这项技能是必不可少的。尽管 SM 是团队中的一员，但他也应该承担起辅导和鼓励团队成员改进与合作的职责。没有谁比 SM 更了解每一个团队成员的优势和缺点，他/她必须能够帮助个人最大限度地为团队做出贡献。
- 尽管 SM 不需要成为技术专家，**但是如果他们掌握团队所使用的核心技术**，或者至少对团队运作的技术环境有良好的认知，这会有助于他们的工作的顺利进行。这也意味着他们应该对所处环境、关键角色，以及出现问题时该做什么、与谁进行沟通等内容有着良好的认知。

检查哪些 SM 具备同时工作所需的硬技能，是相对容易的。

但是其他技能则没那么容易预先检查。

- **一名优秀的 SM 是快速的学习者**，他们会以很快的速度掌握所需的软技能，并在特定的环境中发挥作用。
- 只要人们作为一个团队进行工作，就会有冲突。**SM 必须能够促进冲突的解决**，让团队作为一个整体而不是各自为战。
- 如前文所述，**SM 是仆人式领导**，他们以身作则、愿意奉献，并做好自己该做的事。仆人式领导就是把团队需要放在个人诉求前面，同时还会帮助他人发挥最大的才华。

总之，优秀的 SM 需具备以下品质：

- 有责任心
- 谦逊
- 追求协作
- 尽职尽责
- 有影响力
- 知识渊博

5.7 Scrum 事件概述

谈论 Scrum 总是一个先有鸡还是先有蛋的问题——如果你谈论事物是如何工作的，那么就会有关于角色或职责的问题。如果你先谈职责，那么就会有关于事物是如何工作的问题。

在深入研究事物的工作原理之前，先讨论角色是明智之举，因为这可能是“两害相衡选其轻”。

通常情况下，Scrum 中预先定义的活动称为**事件**，使用的或生成的资源有时候称为**工件**。

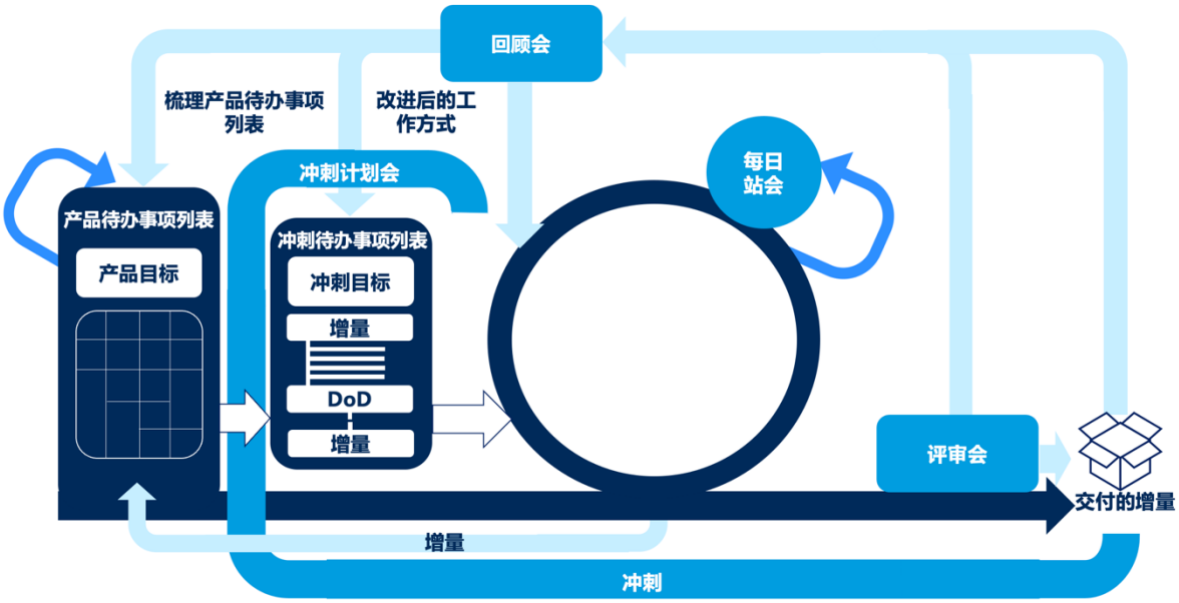
干系人的需求被收集并记录为用户故事，这些用户故事组成了一个或多个产品待办事项列表（下简称 PBL）。这些将在后面更详细地描述。

每个产品都有自己的产品待办事项列表并且产品经理为 Scrum 团队的产品结果价值最大化负责。产品负责人不是项目经理，但最好是产品或相关产品组合的业务负责人。

产品待办事项列表中的故事是由产品负责人整理完成的，产品负责人将与 Scrum 团队中的其他成员一起合作，根据它们的优先级在最短时间内交付产品待办事项列表中的故事。这项工作将持续进行，直到所有收集到的需求都得到满足。

针对需求的交付是在简短的、小项目式的、计划好的迭代事件、被称为**冲刺**中完成的，并且该过程是反复进行的。

图 3：Scrum 中所有内容的概要视图-浅蓝色代表事件



图片由 EXIN 基于以下内容创建：Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value*. Amazon Digital Services LLC – Kdp Print Us.

Scrum 团队会咨询他们的产品负责人并选择他们在规定时间段内（冲刺）可以完成的需求。设定的时间段通常对所有的冲刺都是一样的。冲刺是具有固定时限（又称为：**时间盒**）的事件，通常是一到四周的时间。

一些组织为所有的冲刺都设置了固定的时间周期，比如四周，而另一些组织则会根据需要有所调整。一般来说，冲刺的时间长度是没有明确规定的，但是在初始阶段，定义固定长度的冲刺是有价值的。

作为起点，建议每个冲刺采用固定的四周时长。在规划和执行工作的时候，使用时间盒够获得更好的可预测性，使得我们可以更容易地理解冲刺的可交付成果以及产品或功能的发布日期。

当反复出现的问题使你无法完成时间盒少于 4 周的冲刺时，通过咨询产品负责人，您可以延长时间盒的长度，但延长后的长度不得超过 4 周。

Scrum 团队自主运作。他们选择并规划自己的工作，并对可交付结果共同负责。如果冲刺失败[19]了，则每个人都失败了。

我们已经指出，需求在 PBL 中被定义为用户故事，用户故事也定义了团队要实现的结果。关于用户故事的更多细节将在后面介绍。

Scrum 团队将需求拆解为能够在一个冲刺中交付的较小的需求。作为**冲刺计划**的一部分，Scrum 团队会将较小的故事拆解为任务，团队成员可以从中选择最合适任务的去完成。

作为冲刺计划的一部分，需要为冲刺中交付的每个增量定义完成的定义（下简称 DoD）。DoD 可作为一种保证措施，用于检查计划和承诺的工作是否已完成和交付。请注意，一个冲刺必须至少交付一个增量。

冲刺计划也是有时间盒的，冲刺计划 4 周的冲刺不应该超过 8 小时。

然后团队开始跟踪交付的进度，这主要通过**每日站会（Daily Scrum）**来实现的。在这不超过 15 分钟的时间盒中，团队成员报告进度、讨论困难，并告知其他成员每个人正忙于处理的工作。有时候依赖关系还意味着团队成员之间的工作应该在每日站会或其上层结构中进行密切协调。其中有一种名为 Nexus 的方法将在后面详细介绍。

在冲刺结束时，团队演示那些可能对客户或用户可用的和可部署的产品或功能。当产品完成了以下测试：功能测试、集成测试、性能测试和可用性测试时，可运行或可部署的产品或特性有时被称为*可交付的产品*。这个事件被称为**冲刺评审**，对于一个为期四周的冲刺而言，这个事件不要超过 4 个小时。

一个可交付的产品能够被用户使用，并为用户和企业带来价值。它可能不包含所有被要求的功能，但是必须验证它是可运行的并提供了预期的成果。一个可交付的产品可以是一个或多个增量，只要它是在一个冲刺中交付的。

在冲刺评审之后，团队评估他们自己在冲刺中的表现，看看他们将来如何做得更好。用于自我评估的事件称为**冲刺回顾**。冲刺回顾也是有时间盒的，为期四周的冲刺的回顾会需要 3 个小时。

尽管没有被正式定义为 Scrum 事件，但 Scrum 团队经常会花费一定的时间（通常也是具有时间盒的），来帮助产品负责人梳理 PBL。

请注意：产品待办事项列表优化 refine 过去被称为梳理 groom，但由于在某些文化中与梳理相关的负面含义，该术语被改为“优化”。本书依然使用“梳理”。

产品待办事项列表梳理有助于确保产品待办事项列表中的条目在下一个冲刺开始之前处于正确的顺序。

请注意，尽管团队通常用特定的时间来进行产品待办事项列表梳理，但这是一项持续的活动。因为团队始终在持续学习，并且可以发现新的依赖关系。一旦发生这种情况，产品待办事项列表就会立即用最新信息或见解进行更新。

---

[19] 冲刺失败一般指迭代结束时未能完成冲刺目标，此时 Scrum 团队每个人都对此负责。——译者注

请注意，本章的意图不是全面描述每个冲刺事件，而是简述了 Scrum 中事件的流转过程。下一章将详细介绍 Scrum 事件。

5.8 Scrum 事件

在前面的概述中，提到了五个 Scrum 事件。他们是：

- 冲刺
- 冲刺计划
- 每日站会
- 冲刺评审
- 冲刺回顾

我们将在本章中简要描述每一个事件。请注意，Scrum 事件是非常简单和易于理解的。

5.8.1 冲刺

冲刺是 Scrum 的核心，在这里将创意转化为价值。

它们是固定时长的事件，为期一个月（4 周）或更短，以创造一致性。前一个冲刺结束后，下一个新的冲刺紧接着立即开始。

实现产品目标所需要的所有工作，包括冲刺计划、每日站会、冲刺评审和冲刺回顾，都发生在冲刺中。

在冲刺期间：

- 不得做出危及冲刺目标的变更。
- 不能降低质量与质量需求。
- 产品待办事项列表按需梳理。
- 随着团队对冲刺中的产品待办事项列表条目进一步了解，可以与产品负责人就工作范围加以澄清和重新协商。

在向产品和冲刺目标迈进时，通过检视和适应活动来确保可预测性。当冲刺太长时，冲刺目标可能会失效（因此与产品待办事项列表中的变更不一致）、复杂性可能会上升，同时风险可能会增加。可以使用较短的冲刺以产生更快的学习周期，并限制风险。你可以把冲刺视为一个短期的项目。

有许多工具可以用来预测进度，如燃尽图、燃起图或累积流图。尽管这些工具被证明是有用的，然而他们并不能取代经验主义的重要性。在复杂的环境中，未来将会发生什么是未知的。只有已经发生的事情才能用来做前瞻性的决策。

如果冲刺目标已过时，那么就可以取消该冲刺。只有产品负责人有权力取消冲刺。

然而，当团队意识到冲刺由于其他原因而无法完成时（通常是当 Developers 注意到由于预期外的复杂性而无法在时间盒内完成工作时）最好修改产品待办事项列表的顺序，以检查哪些工作应该优先被考虑。

5.8.2 冲刺计划

冲刺计划通过安排冲刺中要做的工作来启动冲刺。最终的计划是由整个 Scrum 团队协作创建的。

产品负责人确保与会者准备好讨论最重要的产品待办事项列表条目，以及它们如何映射到产品目标。Scrum 团队还可以邀请其他人参加计划会，以提供建议。

冲刺计划处理以下主题：

为什么这次冲刺有价值?

产品负责人提议产品如何在当前冲刺中增加其价值和效用。然后，整个 Scrum 团队共同制定一个冲刺目标，用以沟通当前冲刺对干系人有价值的原因。冲刺目标必须在冲刺计划结束之前最终确定。

这次冲刺能完成什么?

通过与产品负责人讨论，Developers 从产品待办事项列表中选择一些条目放入当前冲刺中。Scrum 团队可以在讨论过程中梳理这些条目，从而增加理解和信心。

选择在冲刺中可以完成多少工作可能会充满挑战，特别是如果 Scrum 团队是敏捷与 Scrum 新手的话更是如此。在这种情况下，建议要慎重的与干系人分享速率期望；然而，充分了解他们的期望有用的，这样能够将他们的期望进行管理。

Developers 对他们过去的表现、即将到来的冲刺的容量（capacity）以及对 DoD 了解得越多，他们对冲刺预测就越有信心。

如何完成所选的工作?

Developers 需要为增量选定的产品待办事项列表条目规划必要的工作，这些工作需要满足增量 DoD。这通常是通过将产品待办事项列表条目分解为一天或更短时间能完成的较小工作条目来达成的。这些分解由 Developers 自行决定，没有人告诉他们应该如何将产品待办事项列表条目转化为有价值的增量。

冲刺目标、当前冲刺所选择的产品待办事项列表条目以及如何交付它们的计划合并称为冲刺待办事项列表。

冲刺计划是具有时间盒的，对于一个月的冲刺来说最多 8 个小时。对于更短的冲刺，冲刺计划所需的时间通常更短[20]。

5.8.3 每日站会

每日站会的目的是检查冲刺目标的进度，并根据需要调整冲刺待办事项列表，以及调整即将进行的计划中的工作。

每日站会是一个属于 Developers 的 15 分钟的事件。为了降低复杂性，它在冲刺的每个工作日都在同一时间同一地点举行。如果 PO 或 SM 正在积极处理冲刺待办事项列表中的条目，则他们将作为 Developers 参与其中。

---

[20] 一般根据冲刺长度不同而等比例变化，比如 2 周的冲刺所需的冲刺计划的时长为 4 个小时。——译者注

只要 Developers 每天都在关注冲刺目标的进展，并为第二天的工作制定可行的计划，他们就可以任意选择想要结构和技术来举行每日站会。这创造了 Developers 的关注焦点，并提升了自我管理的能力。

每日站会可以改善沟通、识别障碍、促进快速决策，从而消除了对其他会议的需求。

每日站会并不是唯一的允许 Developers 调整计划的时间。他们可以在任何时间碰面，就调整或重新规划冲刺剩余工作进行更加详细的讨论。

在 Scrum Guide 的早期版本中，每个团队成员都应该回答三个特定的问题。它们是：

- 我今天在做什么？
- 我昨天完成了什么？
- 我遇到了哪些阻碍我完成工作的问题？

很多人认为这个清单是唯一应该被问到和回答的问题，这也是为什么将它们从 2020 年的*《Scrum 指南》*中删除的原因。

Scrum 团队应该询问、回答和讨论任何相关的问题，以确保讨论尽可能全面以及工作流程正常运行。

这完全取决于团队如何利用这 15 分钟来更好地制定规划和执行当天的工作，以及如何处理阻碍和问题。

5.8.4 冲刺评审

冲刺评审的目的是检查冲刺的结果并确定未来的适应性。Scrum 团队向关键干系人展示他们的工作结果，并讨论产品目标的进展情况。

在冲刺评审期间，Scrum 团队和干系人将评审在这次冲刺中完成了什么，以及环境发生了何种变化。根据这些信息，与会者可以就下一步的工作进行协作。产品待办事项列表也可能会进行调整以满足新的机会的要求。评审会是一个工作会议，Scrum 团队应该避免将其内容仅限于展示。

冲刺评审是冲刺的倒数第二个事件，它也是具有时间盒的。对于一个月的冲刺来说，最多为 4 个小时。对于更短的冲刺，冲刺评审所需的时间更短。

5.8.5 冲刺回顾

冲刺回顾的目的是规划提高质量和效能的方法。

Scrum 团队检视最近一次冲刺中有关个人、交互、流程、工具以及团队所定义的 DoD。被检视的元素通常随工作领域而变化。导致他们误入歧途的假设被识别出来，并探究它们的根源。Scrum 团队讨论在冲刺期间哪些东西进展顺利、遭遇到哪些问题，以及这些问题是如何被解决的或未被解决。

传统意义来说，团队具体讨论三个方面。和每日站会一样，并不局限于这些主题。然而，在回顾会上问这些问题可能会有所帮助：

- 我们要停止哪些行为？
- 我们要继续哪些行为？
- 在未来的冲刺中，我们有需要改进哪些内容？

请注意，这些问题只会有助于谈话的有效开展，Scrum 团队应该识别出对提高效能最有帮助的变化。最有影响的改进应该尽快实施，甚至可能被其添加到下一个冲刺的冲刺待办事项列表中。

冲刺以冲刺回顾的结束作为结束标志。它也是有时间盒的，通常一个月的冲刺最多 3 个小时，对于较短的冲刺，冲刺回顾所需的时间更短。

5.9 Scrum 工件

本文将详细介绍所有提到的工件，即：

- 产品待办事项列表，后续简称 PBL
- 冲刺待办事项列表，后续简称 SBL
- 增量

虽然没有定义为工件，但这些概念与 Scrum 工件密切相关：

- 产品目标
- 冲刺目标
- DoD
- 用户故事（以及这些用户故事的变形，包括史诗故事和特性）
- 任务分解

第二个列表和三个工件之间的关系可以描述为“用以创建或维护工件的承诺”：

- 产品待办事项列表的承诺是产品目标
- 冲刺待办事项列表的承诺是冲刺目标
- 增量的承诺是 DoD

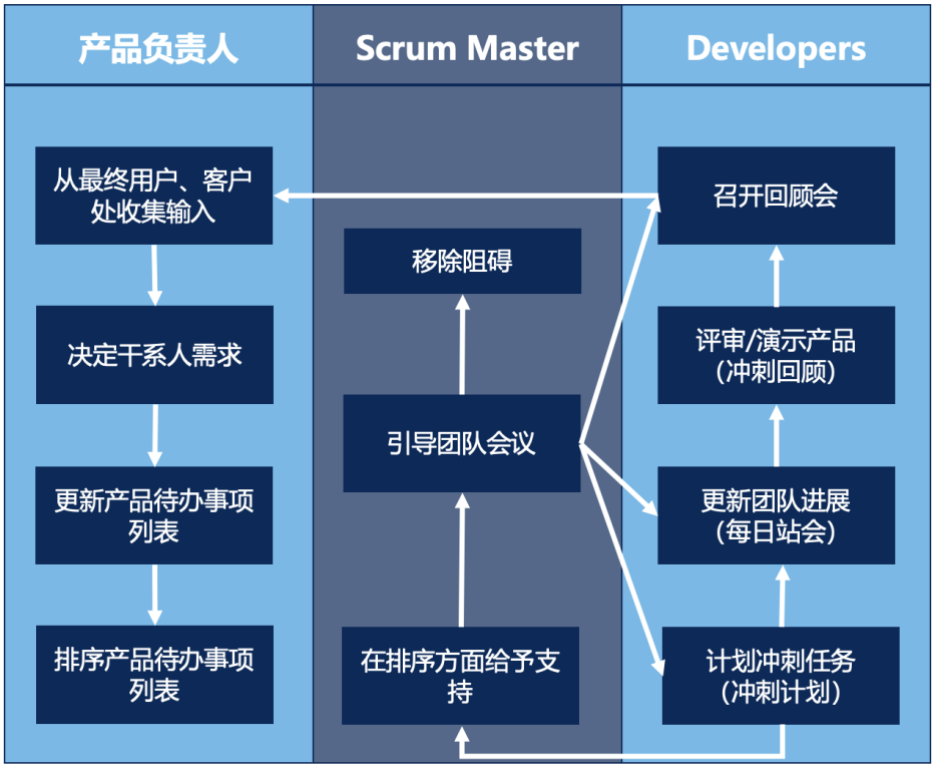
尽管没有承诺，但是用户故事被用来描述产品待办事项列表条目，并且在冲刺计划中，它们被分解成的相互关联的任务。

您可能已经意识到 Scrum 中定义的角色非常简单。如果再写一章关于 SM 或 PO 需要做什么，然后只描述他们在事件中的作用，或者在工件的使用或创建中的作用，或者在实际操作中的一些其他活动，那将是毫无意义的。

相反，本书其余部分的做法是讨论一个概念或工件，与此同时我们会描述 Developers、SM 和 PO 在特定事件中的角色，或者将其作为所讨论的概念或者工件的一部分。

总之，下图是关于 Scrum 中角色和相关活动的快捷提示。这不是一个流程图，而是 Scrum 中角色和活动的关系图。

图 4：高亮显示 Scrum 活动之间的关系（这不是流程图！）



图片由 EXIN 基于以下内容创建：Botha, J. (2021). *[Courseware]*. GetITright.

6 Scrum 团队其他的活动

6.1 组合，产品，路线图

严格来说，Scrum 没有针对组合管理问题提供过指导性意见；Scrum 提供的指导性意见始于产品层面。而实际上，产品也不是凭空出现的。

任何组织都存在一系列要做出关键决策的事件周期，以及与其关联的规划，以便于确保决策符合组织的环境，并且确保做出的决策事先已知情的。

在组织内还需进行一系列规划活动和相关的决策。先在战略层次开始规划，然后是产品管理方面，最后是运营需要完成的工作方面（Developers 层面）。注意，运营环境可以划分为*运营管理*（在进行的活动）和*运营变更*（项目）。Scrum 主要关注后者。

在实施或者扩展 Scrum 的时候，有必要去理解事件周期的基础信息，或者从战略到运营一系列层次。在 Scrum 环境中大概是：

- 组织目标
- 组合决策
- 待办事项列表创建
- 待办事项列表梳理
- 或者规模化交付规划
    - Nexus 或者 Scrum@Scale，甚至是发布计划）
- 冲刺规划

请注意，虽然讨论决策是有帮助的，但是在敏捷环境中这些决策并不是神圣不可侵犯的，改变决策或者改变计划从来都不是问题（响应变化**高于**遵循计划）。确保时间没有浪费在不必要的规划上。在每一个系列层次上做最少的规划。当你对环境和客户需求有更多的了解（即经验主义）或者随时间推移对现实的认识也有变化的时候，再来改变计划会容易的多。

美国前总统 Dwight D. Eisenhower 曾经说过：在准备每场战斗的时候，我总是发现计划并没有多大帮助，但是规划却是不可或缺的。

请记住 Eisenhower 的话；之所以规划是不可或缺的，是因为它是我们学习和理解过程的一部分。所以把注意力集中在规划上而不是计划上，鼓励变化和学习，让改变计划变得尽量简单。

6.2 组合规划

对于许多大型组织来说，需要被完成的工作是在一个更广的组合内进行管理的。讨论中的组合可以围绕产品、系统、价值流、供应链、投资，甚至是项目。

组合管理和规划的目标是确定哪种产品支持组织的目标和目的。

基于组织的优先级，我们可以理解组织需要以什么样的顺序来研发产品以及产品在什么样的条件下能满足组织需要。

组合规划应该涉及广泛的干系人范围，也应该涉及产品负责人。组合规划周期较长，可以是未来 6 到 12 个月。

因此在实践中，我们应该关注的问题是：组合规划的产品将如何组织向其目标和目的靠近？坚持敏捷方法，组织角度的组合管理可以通过创建一个包含该组合内每个产品的产品路线图的组合列表来表现。

6.3 构想你的产品

构想产品的起点是去构想潜在产品的要素，以及生成关于如何构建产品的笼统的想法。产品负责人和干系人一同在高层级上构想一个产品，其规划周期可超过一年。但由于世界变化太快太频繁了，所以如果可以，还是推荐使用较短的规划周期。

参与者应考虑产品的目的或目标，并确保讨论内容始终不脱离产品将要助力的组织目标或目的。

*构想的主要输出*应该是定义一个产品目标，形成一个高层级的产品待办事项列表，并通过定义高层级的产品路线图（一个活的、持续演进的文档）创建一份关于待办事项列表将如何转化成价值的视图。这些输出将成为更高层级组合规划的输入。

最初的高层级 PBL 为产品负责人识别整体和大致的功能，或者识别干系人期望从产品中获得的收益提供了一个明确的起点，有助于产品负责人开展详细需求的收集工作

虽然产品路线图是沟通产品目标的一个有效的工具和机制，同时还具有体现产品将如何构建和交付的增量属性，但实际情况也常常和最初设想的有很大差异。

6.4 产品和产品目标

Scrum 团队应将产品目标作为一项规划参考，因为产品目标描述了产品的一种未来状态，指出了哪些高层级组织目的和目标是产品支持的。产品目标是 PBL 的一部分，通常在开始定义 PB 之前得到书面落实，它还是判断添加到 PBL 的事项是否会使组织向产品目标更加靠拢的主要"检查项"。

要定义产品目标和 PBL，首先需要定义在 Scrum 中产品到底是什么。

Scrum 把产品看作是向某人交付价值的方法或者工具。这意味着我们不仅要理解哪个"价值"是产品交付的，还要知道需要交给什么人。

设定清晰的边界是有必要的。了解谁是干系人、他们有哪些角色、什么构成了干系人所需的价值，是一个很好的出发点。

在 Scrum 中，产品和服务没有区别；产品既可以是有形的实物也可以是无形抽象的。

产品目标是 Scrum 团队的长期目的。每个冲刺目标都是向产品目标迈进的一步，团队必须完成或者放弃一个目标才能继续下一个。

6.5 产品目标和业务价值

产品目标是构想的产品的输出和产品所创造的价值，是一个关于产品是什么的高层级说明。

但是业务价值如何跟产品目标产生联系？

业务会制定组织必须实现的目标和目的来达成组织战略。有时候，组织组合管理中的产品能推动组织实现这些目标。特意使用了"有时候"这个词，因为组织中还有其他实现设定目标的途径。

所以，组织目标和产品目标的区别是前者与商业战略相关，后者与产品管理相关。

Scrum 中的*产品目标*之所以不同于该术语在通用产品管理中的用法，是因为它在 Scrum 中的用法更加具体。在早期 Scrum 版本中，有一个关于*产品愿景*的说法。相比于通用产品管理领域的通用用法，产品目标这个术语的用法更接近于这里说的愿景。

产品目标是关于产品的长期目标或者产品未来状态的一个简述。可以说产品目标是产品的 WHAT，有时候以一个 WHY 为补充。

也可以说，产品目标反映了 Scrum 团队对 PBL 的总体承诺并提供了团队的关注点。

使组织实现其战略目标本身就是有价值的，这是产品目标的主要关注点。

不过，重要的是要考虑到，产品通常会赋能组织/组织的客户，所以对于业务来说，产品在各个方面都是有价值的。产品经常会创造内在价值和外在价值两种价值。内在价值改进组织流程和工作方法，外在价值促使终端用户执行以前无法进行的工作或将以前执行过的工作做得更好。产品通过多种方式创造价值，同样一个产品甚至是特性可以为该产品的不同用户创建不同的价值。作为产品中有价值的增量，交付价值远远超出产品目标所定义的范围。

虽然增量交付价值与总体产品目标相关，但并不是所有的产品价值都在产品目标描述中体现。

应该注意，我们不能通过收集产品创造的所有类型的价值而把产品目标定义得过于庞杂。一个庞杂的产品目标会变得没那么重要，因为从中不容易找到焦点。

可将产品目标视为产品支持的最重要战略目标或者目的的描述。

就这一点而言，我们要问的下一个问题是：什么是价值或者有价值的?

价值是由产品或者功能所解决问题的对象来决定的。尽管价值是一个非常主观性、不易定义的术语，还是需要去努力发掘其定义。

造成这一切的主要原因是，创造和交付价值都消耗了资源，因此产生了有形成本。进而导致除非能回答下面的问题，否则关于价值的问题是无法回答的：从货币角度，创造出的价值是否高于创造价值过程中产生的成本?

这就是价值在敏捷环境里变得有趣的地方。

6.6 度量实际价值

不要期待传统的财务比率指标能够度量通过敏捷所获得的收益或者产品的实际价值。

相反，我们建议将关注点放在改进和领先指标的度量数据上。财务报告几乎不会对组织绩效产生积极的影响，但是关注日常绩效可以做到这一点。

在很大程度上，传统的工业化世界观推动了传统财务指标的构成和使用，而这与敏捷环境不相适应。但是在 20 世纪 90 年代组织开始接受精益时，挑战很快显现出来。

为什么在敏捷环境中传统财务比率会出现问题，且更易产生误导?

尽管某些证据表明一些财务比率的使用可以追溯到公元前 300 年，但投资于公司作为一项业务的激增凸显了财务比率的作用。他们解决的问题很简单。通过比率，投资者可以比较对不同公司的投资，这些公司在组成、行业、风险和市场等方面存在的巨大差异。通过比率，投资者可以考虑是否应该向一家公司投资，比如一家印度制造木箱的公司、一家美国汽车制造商或一家欧盟金融机构。研究关键比例能让决策更轻松。

本文讨论的比率包括所有传统的会计比率：

- **流动性比率**度量一个公司偿还短期和长期债务的能力。
- **杠杆比率**度量债务资金总额。
- **效率比率**（也称为财务营运能力比率）度量公司资产和资源的利用情况。
- **获利能力比率**度量一个公司创造与利润、资产负债表、运营成本和资产相关的收入的能力。
- **市场价值比率评**估公司股票的价格。

在这里并不是要表达在敏捷环境中所有传统比率都是无用的。可是如果按照一个月，六个月，一年，五年，甚至十年来度量，很多传统比率反映的情况会与现实截然不同，这一切都取决于观察的角度。

传统会计实务和随之生成的财务比率的问题在于财务人员认为财务报表周期是有开始和结束时间的。对于瀑布型项目来说使用传统度量标准和指标也很简单。毕竟，瀑布型项目的定义中就包含一个开始时间和一个结束时间。在这个情况下，通过对比成本和实现的收益来计算项目的投资回报（RoI）是很容易的。

敏捷环境看待业务绩效的方式是完全不同的。

敏捷环境更看重持续改善所有的绩效指标，而这没有起点和终点。因此，大部分传统的财务比率和会计实务对于敏捷或者精益来说都不适用。

实际上，使用传统比率可能会带来短期的改善而长期的结果却是负面的。在敏捷环境中建议进行长期的改善，即使会造成短期的负面结果。

在敏捷组织里考虑会计实务时，你可以有效地运用从精益中吸取的经验教训。公司很快发现需要的是完全不同的财务实务、一个涉及组织更广系统视角的会计实务，并且还会意识到"良好财务绩效"是由什么构成的。产生这种意识的时刻，即意味着精益会计的诞生。

将投资回报（RoI）、内部收益率（IRR）和净现值（NPV）作为度量敏捷投资决策指标所面临的挑战会变得很棘手。

以下是旧的假设不再适用的一些原因：

- 在敏捷中**没有项目开始或者结束日期**。实际上，纯粹的项目是否存在是有争议的，而是交付价值的短增量。
- 由于需求被允许甚至被鼓励随着时间推移而变化，对于随时间推移要做什么并**没有预先的或仅有有限的认知**。

若作为敏捷环境中项目投资决策的指标，RoI、IRR、NPV 将会演变成充满问题的指标。因此需要一套不同的度量指标。

更复杂的是，传统意义上只是一项运营活动的持续改进，将成为冲刺的一部分，从而使"运营项目"和"运营所处环境"之间的界限更加模糊不清。

与精益环境（来自精益会计）一样，目前最合适的度量成功的标准是：

- 改进工作流
- 改进或者保持质量
- 改进后的绩效（交付，交付时间，生产力）

以及随之产生的：

- 改进的盈利能力

精益（敏捷）会计的关注点是一个广泛的、系统化的视角。你可以通过一个特定的窗口报告改善情况以纠正错误，但重要的是要理解这些报告是临时的度量，它并不一定反映公司长期的健康状况。

这种观点的改变意味着向一个部门或一个项目提供资金的做法不再有意义。组织的收入与产品的绩效有直接关系，所以在项目/变更和运营都需要资金的情况下，要更加谨慎的为产品提供资金。你必须考虑创造、销售、服务和维护产品的整个价值流，它是以端到端的形式存在。这种系统化视角与会计们在大学里的所学是大相径庭的！

新的精益/敏捷的 RoI 评估方法更加全面。为了有效地度量投资绩效，有必要将整个价值流中消耗资源的所有事项的总和，与将产品销售给客户而获得的收益进行对比。与绩效相比，报表必须成为一个连续的时间切片实例，而不是将报表当作一系列不相关的实例反映着公司绩效。

你可能会争辩说，交付产品的价值流总成本和获得的收入相比，实质上还是 RoI，是的，的确。但是将来这对你有什么帮助？用过去的 RoI 来度量一个产品未来 RoI 是非常不可靠的。

RoI 这样的比率的缺点是它们是滞后指标，即它可能会告诉你，没有达到预期的投资回报，但作为纠正错误的措施却发挥不了作用。正因为这个原因，精益会计把改进作为绩效的领先指标。

有趣的旁注：丰田的目标之一是以每年 2%的速度全面改进公司的一切。虽然目标听起来很小，但是从 1950 年开始丰田就已经实现了该目标并且在整体上产生了巨大的影响。

如果在这个月、上个月、再往前一个月都有改进，那么你就可以顺利实现对 RoI 的提升。如果在这个月还没有获得改进，你需要立刻采取纠正措施。

在 2012 年到 2016 年之间，敏捷联盟（Agile Alliance）成员开展过多个项目以解决敏捷会计标准化问题，但从该倡议的输出来看，他们基本上没有将敏捷性作为一种综合的、全价值流的、持续的方法来评估敏捷环境下的财务表现。

当前，精益会计是在敏捷环境中评判投资决策最合适的方法，不过它需要你对什么是会计，以及如何在组织中执行会计的理解有相当大的转变。

因此，上述的一切的结论是什么呢？

将关注点放在作为改进和领先指标的度量上。财务报告几乎不会对组织绩效产生积极的影响，但是关注日常绩效却可以。

6.7 管理产品待办事项列表

没有一个合格的产品待办事项列表，Scrum 是行不通的。产品待办事项列表是所有 Scrum 规划活动的心脏，也是 Scrum 团队唯一的真相来源。它体现了团队需要完成的所有工作以及其相对优先级。如果任务不在产品待办事项列表内，需求和相关的活动自然也不会存在。

产品待办事项列表不仅描述了客户的需求，而且描述了所有隐藏在产品背后的技术任务以保证客户需求得到满足。所以产品待办事项列表可以描述为一个将项目中待交付的功能性需求和非功能性需求进行了优先级排序的列表（或者说下文将会出现的排序列表，）。当某个需求实现后，相关的条目就从产品待办事项列表中移除。

一些公司也会把针对需求的待交付任务纳入到产品待办事项列表中，比如规划，资源计划以及其他准备步骤，然后再开始项目工作。

PO 是产品待办事项列表所有者，并为管理那个具体的产品待办事项负责。他们的任务是确保待办事项最新且有序的。他们在 Scrum 团队的帮助下完成该项工作。

一个普遍的做法是保证产品待办事项列表满足四个特定的标准。该标准的概念是由 Mike Cohn 发起定义的，如今仍在 Scrum 社区中广泛使用。它由首字母缩写 DEEP 组成，分别代表：

- 详略适当的（**d**etailed appropriately）
- 估算过的（**e**stimated）
- 涌现的（**e**mergent）
- 排序过的（**p**rioritized）

已有迹象表明 Scrum 在回避使用排列优先级（Prioritized）这一术语，因为在不同的上下文和组织中该术语代表着不同的含义。Scrum 倾向于使用术语**有序的（Ordered）**，那么在 Scrum 环境中 DEEP 也就变成了 DEEO。

- 详略适当的（**d**etailed appropriately）
- 估算过的（**e**stimated）
- 涌现的（**e**mergent）
- 有序的（**o**rdered）

6.7.1 详略适当的

尽管产品待办事项列表是把需求分解成具体活动的地方，需要足够详细，以便能够对完成与需求相关的工作需要多长时间有一个基本概念。

需求通常被记录成故事，不过故事有着不同程度的细节。

图 5：随着故事被移到产品待办事项列表的顶部，详细程度也逐步增加，大的故事被分解到较小的（更可行的）故事里

图片由 EXIN 基于以下内容创建：Botha, J. (2018). *Scrum Masters and Product Owners [Courseware]*. GetITright.

高优先级的事项被拆分到足够具体的程度之后，才可以判断满足相应需求所需的工作量和技能。

因此，需要把在下一个或者两个冲刺中要实现的重要事项在产品待办事项列表的梳理中和冲刺计划中，将其分解到正确的粒度。

这意味着随着项目的开展，把故事或者需求分解到更加详细的程度的工作是持续的。但只在必要时才这么做，并不需要提前分解所有的故事或需求。因为当把故事首次添加到产品待办事项列表的时候，我们甚至还不确定该事项和所有相关需求都是重要的，甚至是否会保留在待办事项列表都犹未可知。重要性低的需求经常会从待办事项列表中移除，甚至会被后来出现的其他需求所取代。

在敏捷中，工作量不应浪费在不被使用的事情上。这就是通过做"刚刚好"的工作来获得成功的含义。

史诗故事和特征都是帮助描述故事的大小和复杂度的**互补性做法**；Scrum 指南没有对不同的故事类型进行区分。

6.7.2 估算过的

产品待办事项列表中所有的事项必须得到过一定量的、针对完成故事所需工作量的估算。显然，故事越庞大，估算的难度越高。所以史诗的估算结果准确性较低，好在影响不大——毕竟史诗故事通常没有那么重要。

估算是能够进行冲刺计划的必要活动。在以增量的方式评估完成特征所需的工作量时，DoD 必须要被纳入考量范围。这说明每一个增量都有一个已完成的定义，因为在冲刺中可以有一个以上的增量。

跨功能团队可以给估算任务提供较大的收益，因为他们有能力在估算活动过程中提供一些缺失的信息。跨职能团队具备不依赖外部资源独立完成工作所需要的全部能力。由于至少有一位具备知识和技能团队成员可以完成任务，因此总会有人能够估算任务。

6.7.3 涌现的

涌现的意思是产品待办事项列表并不是关于产品生命周期中应该发生什么的标准性说明。客户需求在产品生命周期中经常发生变更，新的需求经常会被添加到列表中。不过，这并不是把工作条目添加到产品待办事项列表的唯一场景。

伴随着产品的增量的发生，经常会发掘出以前不知道的新需求。他们通常是其他需求运行的依赖项，很多时候不是功能性需求而是非功能性需求。

因此，当我们说产品待办事项列表是涌现的时候，我们真正想表达的重点是"它将会不断逐步演进"。

6.7.4 排序过的

产品待办事项列表中的所有事项都是通过有序排列的，首先要完成的事项会显示在列表的顶部。顺序体现出的是完成故事的重要性、实施顺序，或是对其他重要故事的依赖性。

6.8 产品待办事项列表梳理

产品待办事项列表是一个不断变化的工件，由于事项在持续添加，移除或者变更，因此它需要得到认真的维护。

上文介绍的 DEEO 活动都涉及产品待办事项列表维护的任务。这类活动也被称为产品待办事项列表梳理。

PO 对梳理负责，不过梳理并不是由单个角色执行的活动。PO 会得到 Developers 的协助一起努力完成。他们凭借丰富的技术见解对产品待办事项列表中已有的或者新添加的事项进行重新排序和重新评估。

之所以说 Developers 做出特定的贡献，是因为他们具备技术知识去估算完成一个故事需要多久、去判断一个重要的故事是否依赖于其他故事并且凭借此依赖继承的重要性。Scrum 团队成员也会知道大的故事可以如何分解才能在冲刺中更容易操作。

尽管 PO 对产品待办事项列表梳理负责，但它是一项协作性较高的活动。

虽然梳理不是一项预先计划好的 Scrum 事件，但是 Scrum 团队成员不得不规划梳理的时间。

由于它不是一项计划好的（时间盒限制）事件，所以产品待办事项列表梳理是在有需要的时候进行的，而且通常发生在其他 Scrum 事件进行期间。如果出现问题，比如需求变更、依赖性或对发掘的需求出现误解，梳理就应该即刻进行。

某个程度的梳理也会不可避免地发生在大规模计划或者冲刺计规划活动期间。同样的梳理也会在冲刺评审会和冲刺回顾会期间发生，这是很正常的情况。完成的故事

需要移除，而其余故事需要根据当前已知的信息重新排序。在刚刚结束冲刺的时候，故事甚至会被分解。如果依赖或者新的需求被发掘，新的故事可能会被添加。

每天花时间讨论产品待办事项列表、冲刺待办事项列表和发现的问题，并且可能会成为每日站会的一部分。这也许是个不错的主意。

比较推荐做法是，Scrum 团队在梳理上尽量花差不多 10%的时间。然而大多数团队在产品待办事项列表梳理上所花的时间都比较少。

产品待办事项列表梳理一般包括以下几个活动：

- **需要将新发现的条目添加到产品待办事项列表里。**他们可以是来自客户的新需求也可以是上个冲刺中遗漏的功能和非功能需求和依赖。
- 通过把最重要的条目放在列表顶部来**排列产品待办事项**。
- **需要合理调整产品待办事项**的大小以便更容易地进行冲刺规划和评估。
- **把大或者模糊的重要事项分解**成小的用户故事，便于安排在接下来的冲刺里。
- **梳理重要的条目**（将其描述得更好），让冲刺规划更加简单。
- **注意依赖**，而且记录依赖需要执行的顺序。
- **对产品待办事项列表进行一次检查**确保所有新的事项均已添加进来，执行的顺序已经确定，大小和完成估算都是合理的，不再需要的事项已从产品待办事项列表里移除。

6.9 创建产品待办事项列表条目

正如前文提到的，需求收集在 Scrum 中不是一次性的事件。尽管在项目开始的时候会很有多需求收集工作，但是注意力不应该集中在获取具体需求上，也不应该把很多时间花在把史诗故事分解成细粒度用户故事上。

分解大的故事是一个持续进行的活动，最好在工作结束之前完成。这样，团队可以确保完成的需求是当前最新的而不是两年前的某个愿望。总之，这么做，你可以确保你所关注的正是当前创造最大价值的需求。

在上一节中强调了需求不会保持静态，新的需求不断出现，经常取代旧的需求。所以花太多的时间去分解那些并不会进入后续冲刺的故事实际上是浪费的；你可能做了很多永远不会用到的工作。这与敏捷做“刚刚好的”工作的原则是一致的。

总之可以说，当你第一次开始收集需求的时候，你需要理解需求和产出，而不是你将如何完成这些需求。

思考需求的一个好方法是，首先通过创建一个路线图来定义项目中要做的事情的高层级视图，然后开始思考路线图的每个阶段的需求。

创建路线图也会迫使你去考虑可以马上交付并立刻开始增值的核心功能，之后再关注次要功能和使产品更好的功能。

收集高层级需求可以通过头脑风暴会议完成，所有关键干系人都参与其中。通过确定你想做什么，以及创建一个产品愿景来开始这场会议。你在解决什么问题或者你将解锁什么价值?

接着产品可以分解成高层级的组件，这些组件的开发可以作为产品路线图的基础。

这个时候不能涉及太多细节，当前还只是泛泛而谈的阶段。我们首先要定义最小可行性产品（MVP）应该是什么。所以，重点应该在核心功能上。缺少了核心功能，产品是无法工作的。

请记住产品待办事项列表会随时间流逝而得到梳理，更详细和次要功能也总能在晚些时候添加。

动态系统开发方法（DSDM）谈到过一个叫做 MoSCoW 的概念，将它用在这里会比较合适。

MoSCoW

MoSCoW 是由动态系统开发方法（Dynamic Software Development Method[21]）开发者之一 Dai Clegg 提出的优先级排序方法，并打算用于冲刺阶段的条目排序。在需求发掘过程中使用 MoSCoW 方法非常有帮助，它可以为 Scrum 团队提供有价值的见解。

当定义一个需求时，你会问这个需求是否是一个 ：

- 必须有（Must Have）
- 应该有（Should Have）
- 可以有（Could Have）
- 不会有（Won't Have）

在这种情形下，推荐先把注意力放在“必须有”和部分“应该有”的需求上，因为通常这两类比较难做区分。

必须有和应该有的区别可以定义为以下几项：

- **必须有**的需求（M）是关键的，应尽快实现的，以便交付期待的价值。如果不交付这类需求，项目有可能会失败。
- **应该有**的需求（S）是重要的但可能没那么紧急。一般来说，应该有的需求要在项目结束的时候交付以确保项目功能齐全或者易于使用。
- **可以有**的需求（C）是一些最好有的特征；不会有的需求（W）是明确排除的，因为没有正当的理由将这些事项作为项目的一部分。

6.9.1 分解非功能性需求

在“不要立刻分解需求”的原则下，有一个例外，那就是非功能性需求。

非功能性需求属于解决方案的一部分，也是解决方案中其他需求所依赖的对象，但它不是由客户提出的。最好的情况是你比较了解交付此类需求都涉及到什么，因为非功能性需求是项目其余部分的基础。

因此，一旦获取到非功能性需求，团队总是尽快并尽可能彻底地对其进行分解。

[21] DSDM, 现在也被称为 ABC（Agile Business Consortium）

6.10 收集需求 – 输出和成果

在 Scrum 中，理解用户和客户期待的结果是非常重要的。理解这些成果是 Scrum 从一开始收集需求采用的基本方法。

那么对进入产品待办事项列表设定一个最低门槛能很好的理解哪些干系人将依赖该需求（为识别依赖提供了情景和帮助），以及**什么**是需要的，**为什么**是需要的。

正因为这个原因，用户故事会通过以下格式表达：

作为<干系人角色（Who）>，
我想要<什么样的需求（What）>，
以便于<实现怎样的价值（Why）>。

如果你从已经收集到的需求入手，那么请格外注意。我们建议您不要把已经收集好的需求转换成用户故事。

传统的需求收集方法倾向于定义细粒度的和详细的需求，与以上所描述的方法完全相反。

仅仅把需求翻译或者复制到 PB 里会使列表非常复杂，而且缺少对优先级排序和价值创造的洞察。

如果你需要管理传统的详细需求，建议通过需求分类（与工作分解相反）创建高层级的需求，进而形成更少的事项。这样更易于对需求列表进行排列、排序和估算。

此时可以使用的方法是把所有详细需求都写到便利贴上，根据相关需求的分类创建一个亲和图，然后可以把分类的描述作为粗粒度需求加入到产品待办事项列表中。

使用该方法也会删除或者取消本属于“可以有”和“不会有”优先级的需求。

用户故事和其他需求可以通过多种不同的方式在产品待办事项列表里体现。有的人喜欢预先设置好的格式，有的人不喜欢。使用预先设置好的格式的缺点之一是它会自动迫使人们用某种特定的方式思考需求。

有时候用户故事只是以列表行的形式存在，而在其他时候它们可以包括很多细节，例如干系人、优先级、依赖、相关活动，甚至谁将在冲刺中管理该需求。然而多数情况下，记录所有这些信息是有些冗余的。

以上介绍的用户故事的格式确保了所有必要的需求的信息能在产品待办事项列表条目中得到记录。使用用户故事的格式，你回答了要实现**什么**，**为了谁**，以及为**什么**。同时，**什么**和**为什么**都是性能或者验收标准。

请注意 Scrum 没有规定要通过用户故事的格式定义需求，但是强烈推荐用它来记录需求。

6.11 更多关于用户故事的介绍

用户故事被称为故事的原因是需求干系人不会使用专业术语向你描述需求，而是会以故事的形式描述。这个故事讲述了他们做什么或者要做什么，以及为什么这么做对他们来说是重要的。

最后一点很重要！前文强调过迭代交付不仅仅是为了结果，而且为了帮助干系人频繁且尽可能快速地实现成果。单个故事的原因（Why）部分就是成果。

当讲述一个关于如何和为什么的故事时，它能帮助故事记录人评估故事是否合理，而仅仅记录下需求的传统方式做不到这一点。

如果需求不合理，记录人可以要求澄清，他们也可以询问为什么和如何做。

你具体是怎么做的？你为什么要那么做？你到底为什么要做？如果你能得到同样的结果，通过另一种方式做是否可以？

对话增进理解，而通过理解可以更好地了解如何为干系人（多数情况下，干系人就是用户和客户）创造价值。

图 6：故事和任务卡片的示例

图片由 EXIN 基于以下内容创建：Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value*. Amazon Digital Services LLC – Kdp Print Us.

使用故事的格式记录和理解需求是一种特别强大的方法。

作为<干系人角色（Who）>，
我想要<什么样的需求（What）>，
以便于<实现怎样的价值（Why）>。

你或许会用某个工具来存储需求，但是依然推荐用实物卡片或者便签来记录，并在看看板上反映交付进度。

下图是一个例子。它实际上是撰写本书过程中使用的看板，左侧的列表是产品待办事项列表。在本例中你会发现故事都很简单，并且只记录了成果，这些信息已经足够实现一个产品。在本例中，这个成果就是您手上这本书，这也证实这些信息是*刚刚好*。

即使在这个个人的例子里，创建产品待办事项列表、定义需求、以及梳理需求也是有价值的。

本例中，浅蓝色"便签"是分解后的故事，原本属于其中一个更高层级的史诗，在这里用更大的深蓝色便签表示。每个不同颜色的便签都有自己独特的含义，通常标签还附有补充信息。这个不是常规做法，而是一个高度定制化的工作方式。每一个 Scrum 团队都将随着时间的推移调整他们自己的工作方式。

图 7：产品待办事项列表

图片由 EXIN 基于以下内容创建：Botha, J. (2021). *[Courseware]*. GetITright.

请注意，记录需求只是旅程的开始。需求将被不断梳理直到完成和交付。

6.12 用户故事及任务分解

下面的章节会介绍估算的技巧和任务分解。在这里强调一下，只有知道完成故事所需的交付内容后才能处理这些故事。

进行任务分解和任务估算是冲刺计划的最后一步。SM 和 PO 不应该影响 Developers 的估算工作。估算期间，PO 必须在场以澄清期间任何可能出现的问题。

任务分解不是一个单一的事件，尽管需要在开始冲刺计划或者任何规模化实施规划开始前完成高层级的分解，但仍需在开始冲刺工作之前全面完成。请注意，如果有需要，分解工作仍会在冲刺期间进行，正如产品待办事项列表梳理一样，分解会再次发生直到工作完成并交付结果。

理论上说，完成任务的时间会像冲刺一样长，但是明显这是比较糟糕且不现实的想法。所以出现了下面两个问题：

- 任务要持续多久？
- 如果任务完成时间较长，我们该怎么应对？

如果任务跟每日站会的步调一致，这样总是最好的，因为在站会期间，团队会就当天要完成的内容进行讨论。

然而不可避免的是，你会发现完成某些任务的时间会超过一天，但是如果做进一步思考，其实你应该能够把他们拆分成子任务，并评估每个子任务的完成时间。

将任务分解到过于详细的程度也会影响完成效率。考虑到团队成员的技能组合，原则是任务分解到一个易于被团队成员接受的程度，而不是过于详细的程度。

建议完成一项任务的时间最长不超过 8 小时。有时候，在已确定的用户故事中添加更多的故事是很诱人的，但是不推荐将故事合并放入冲刺中，因为这么做会使故事规模过大且难以管理。

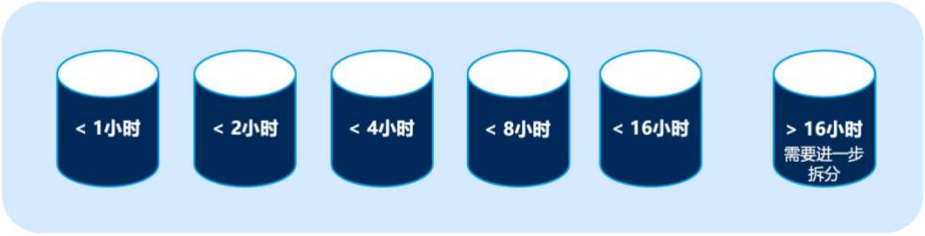
如果你无法把任务分解到 8 小时的水平，可以考虑在团队成员每日站会沟通进度时加入额外的反馈。这可以成为一个很有帮助的做法。假如任务未在一天内完成，团队成员不会说分配给他们的时间有多少用在了这个任务上，而是完成该任务还剩下多少估算的时间。

该做法有两个主要的优势：

1. **推动人们去主动思考工作**并帮助他们学习未来如何更好地进行任务估算。
2. **帮助 SM 更新进度度**量工具来反映了迭代剩余的时间，比如燃尽图。在这里，纯粹主义者会说，燃尽图并不反应已经完成了过半的任务。理论上这个说法是对的，但是反映未完成任务的优势是帮助团队去更好的计划，同时也可以成为忙于任务的团队成员的激励，毕竟没有人愿意说"我今天什么也没完成"。

当进行规划和任务分解的时候，你可以使用以下的"水桶"来评估任务持续时间。

图 8：将估算的工作量分配到水桶系统中



图片由 EXIN 基于以下内容创建： Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value*. Amazon Digital Services LLC – Kdp Print Us.

6.13 创建和维护产品待办事项列表和产品路线图

在上一章节中，我们讨论了需求收集以及首先着手于粗粒度（高层级）需求的必要性。

《Scrum 指南》将产品待办事项列表描述为包含了所有已知产品需求的有序列表。它是对产品所做的所有更改的单一来源，这些修改都是为最终满足产品目标和组织目标而存在。

注意"所有"这个词。确保你还把非功能性需求放在了列表里也是非常重要的，甚至推荐把在冲刺评审或回顾中发现的尚待解决、不太理想的关键问题也包含进来。

你可以用 4 到 5 个粗粒度用户故事（史诗故事）来启动一个产品待办事项列表，然后按优先级进行分解，然后在每个迭代中依次将其完成。一个经验法则是：Product Backlog 不要超过 100 个事项，真的超过的话最多也就 300 个。

任何超过这个数字的情形都会使产品待办事项列表梳理变得几乎不可能，并使其管理异常困难。

建议遵守 100 条事项的原则。一旦看到数量增多，就要考虑可以把哪些事项从列表中删除。如上所述，在讨论用传统方式收集起来的需求时，再次实践该原则，将详细的优先级低的事项合并到史诗中。

如果你从 4 到 5 个高层级需求开始，也可以创建一个产品路线图，也就是给每个需求分配一个评估过的执行顺序就可以了。

但是大部分情况下，事情并没有这么简单。

为了创建路线图，你需要把高层级需求按照主题分组，并把这些主题作为创建路线图的指导方针。通常情况下，主题可以描述为与产品或者产品类别、商业用途（包括支持类流程）或者功能相关的一组需求。

然而现实是，在复杂项目中，根据主题分组创建产品路线图，尤其根据产品、产品类别或者商业用途来创建路线图并不是最佳的方式。基于多个相关功能去创建主题反而是更加简单和有效的。

然而这么做也并非总是理想的，因为产品路线图的目的是向更多干系人提供一个关于价值将如何创建的想法。如果没有技术背景，大部分干系人无法根据主题理解产品路线图。只有跟他们正在做的或者想做的有关联它才有意义，而这也就是需求。

图 9：产品路线图示例

图片由 EXIN 基于以下内容创建： Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value*. Amazon Digital Services LLC – Kdp Print Us.

在定义主题的时候，你要清楚为什么这么做，以及创建主题对你的帮助或者阻碍。时刻问自己：用户和客户会明白我们所说的主题吗？他们会理解哪些功能属于哪个主题吗？

对于如何创建产品路线图并没有一个简单的答案。让我们一起探索一些原因。

产品路线图是用来体现产品的愿景、方向和交付顺序的，并不像传统项目计划或者甘特图一样是承诺的时间线。

产品路线图的目的是通知并更新干系人正在发生的工作，同时给他们一种舒适感。

所以产品路线图也是持续变化的，它们随着关键业务的发展而变，随着 Scrum 团队开始处理事项和确定依赖而变。

产品路线图也经常被用作项目内资源或者可视化沟通工具。除了上述沟通的事项外，团队还需要看到随时间发展产品的进度。如此使用，它就成为一种行动计划和汇报进度的方式。

不要给两种受众使用同一个产品路线图。如果你向其他干系人——尤其是管理层——使用了团队视图，你就把产品路线图变成了对客户的一种承诺。这是一个糟糕的做法，因为不出三个月，路线图可能会与当前的大相径庭。

请注意，Scrum 团队的绩效指标用到的维度，绝不能作为向一般干系人展示的进展或者绩效视图的工具。

6.14 如何对产品待办事项列表条目排序?

Steven Covery 在《高效能人士的七个习惯》中曾说过优先级是既重要的又紧急的，但是与大部分人看待这两个词汇不同，优先级应该通过如下标准判断：

1. 重要且紧急
2. 重要
3. 紧急
4. 不重要不紧急

但这些术语意味着什么?

- **重要**意味着业务将遭遇巨大的价值损失，或者若不能完成或者尽快完成则会产生重大后果。
- **紧急**意味着有人对完成它有既得利益，例如，销售人员承诺一个功能将在下一个版本出现。

可以说这两个列出的标准代表了价值和风险。所以价值和风险是重要性的两大关键决定因素，在一定程度上还是紧迫性的决定因素。

显然，兑现对用户的承诺是相当重要的，但如果这意味着先做了某件事会使组织遭受损失，那么很容易决定先做哪一件。价值和风险比短期承诺更重要。与其做出可能难以实现的"廉价"短期承诺，不如和干系人一起决定待完成事项的价值。

当太多的事项被看作是重要的时候将会构成一个重大的商业风险。那意味着没有什么是重要的。通过这个类比可以帮助干系人理解如何定义产品待办事项列表中事项的重要性以及完成的顺序。

还可以使用**帕累托法则**。产品待办事项列表中只用 20%的重要事项。邀请干系人决定哪些事项属于那 20%。

尽管 PO 最终会分配待办事项列表事项的顺序，最好还是能邀请较多的干系人一同理解和决定优先级顺序。

根据商定的条件，使用分配给事项的相对重要性对待办事项列表进行排序。下一个将要完成的任务应该放在产品待办事项列表的顶部。

另外，请保证列表上方的事项都被合理地分解为细粒度的需求且包含准确的工作量评估。

请注意，还有一些事项很有可能会被划分到下一个冲刺的冲刺待办事项列表中。

你可能会发现在分解工作过程中，有一些细粒度的故事明显是重要的，而有些并不是。这时要重新对产品待办事项列表进行排序，并且放在产品待办事项列表上方的只能是在后续几个冲刺里要完成的事项。其他的事项尽管它们是细粒度，但依然要放在产品待办事项列表靠下的位置。或者干系人可能决定把它们从产品待办事项列表彻底删掉。

接下来梳理待办事项列表中一些相似的需求也是有帮助的，他们或许能带来相似或者完全相同的结果，这些重复项可以很容易地从待办事项列表里移除。在此前提下移除事项时，请确保依赖性和需求得到了完整的匹配和正确的理解。

不要把过多的时间花在非重要事项的排序上。在未确定他们变得更重要之前，这真的无关紧要。

必不可少的事项应该得到很好地理解和记录。如有需要，使用单独的文档来管理产品待办事项列表条目。这个通常是通过对干系人提出更多的问题，或者邀请干系人一同分解史诗中来完成的。

如果不太好给一些非常细粒度的事项排序，你可以把它们归类到一个更易理解的故事或者事项中，并在待办事项列表上分配一个顺序。这个实践很有帮助，因为这让我们更容易理解事项之间的依赖性。如有需要，Scrum 团队在冲刺计划期间可以再次对其进行分解。

每次梳理产品待办事项列表时都需要重新确定优先级。这是保持待办事项列表清晰和明确的最简单的方法。

你也会注意到，第一个迭代的结果不久后就会有“新”需求开始进来。但不要以为他们是新的，其实很多都以史诗的一部分或者更高层的故事存在了。

如果需求比较高且变得更重要，你可能要对待办事项列表事项进行重新排序。

非功能性需求是例外，因为他们通常构成更高的依赖，所以在产品待办事项列表中会占据更高的位置。一旦识别出非功能性需求，应立即对其进行分解，并作为细粒度的需求直接放在列表中。风险也是很好的排序的指标。

为避免之后出现额外的工作，最好尽快处理风险和不确定性。把风险高和不确定性高的事项分类后尽可能把它们置于列表最靠上的位置。

这是验证假设的早期实验，而这些假设本质上是不确定且存在风险的。这么做也契合了 Scrum 的原则之一——使用经验方法。

在 Scrum 中，未处理的风险是团队的主要负担，将来也会拖住团队。如果任何已经完成的工作所依赖的对象最终被证明是不可持续的、不真实的、无法使用的，那么所有相关的努力也都白费了。

当在产品待办事项列表罗列事项时，你可以选择根据依赖项进行分组，或者根据要交付“完成”且对用户可用的事项进行分组。Scrum 规模化方法（Nexus）就是针对这个方法的，下文中将有具体介绍。

注意，不要以为所有相关的故事对于冲刺的输出来说都是必要的。很多时候一个冲刺可以交付一些核心的元素并排除一到两个功能，仍可做到完全可用且交付业务价值。

当讨论冲刺计划的时候，你可能会记得有人建议选择属于必须的、应该有和可以有的功能（MoSCoW 排序），应该有和可以有的功能或许不是 DoD 的必要部分，所以会被用来创建冲刺的“缓冲”。

在引入 Scrum 时经常使用以上这种方法，毕竟谁都不希望冲刺失败。这可能会导致你需要公开的对那些处理组织内部失败的传统方法（惩罚性行动）采取行动。

请记住，为干系人早些解锁价值是目的。我们总是尽所能尽早交付真正的价值。

6.15 与干系人的沟通

对于任何类似项目的环境来说，最大的挑战之一是知道什么被视为有价值、知道什么能创造价值。

然而价值很像审美，它取决于观察者的看法。对一个人有价值的东西对另一个人来说一文不值。

还需要重点注意的是，干系人经常描述一些对他们而言有价值的东西。但是进一步调查后发现，并不是所有的东西都具有相同的价值。在这里，MoSCoW 技术有助于引导对话，并帮助干系人区分什么是绝对不可谈判的（必须有），什么是重要的并能极大地促进效率或生产力（应该有），什么是最好有的并且会让事情变得更好或更容易（可以有），以及什么应该是未来要考虑的但当前并不是关键的（当前的不会有）。

在设计和构建新的产品和服务的过程中，时间是讨论的重点部分，因为一些功能形成了产品或者服务的核心，而其他功能赋能或者增强产品功能。因此，Scrum 团队应该首先构建核心部分，然后再关注下一个最有价值的事情。

使用 Scrum 构建新产品和新服务与 Eric Ries 提出的精益创业非常相像。这是一个迭代过程，需要快速学习干系人对产品或者服务的核心的想法。

Scrum 团队可以定义一个**最小可行性产品（MVP）**，并与干系人确认 MVP 是否是他们能用的且能为他们解锁价值的东西。如果是，那么所有组成 MVP 的故事都应该是细粒度并位于产品待办事项列表的顶部。如果这么做，你会发现产品待办事项列表顶部的所有内容都是必须有的事项。如果是这样，MoSCoW 将如何提供帮助？

当 Scrum 团队梳理产品待办事项列表并寻找可以轻松有效地完成必须有的工作时，使用恰到好处的原则，即使那些工作可能属于应该有或可以有，也可以将其纳入工作范畴。

警告：不建议在和其他干系人讨论功能的时候使用这一原则，因为最终你会产生不切实际的期望。请只在冲刺（或者 Nexus）团队而且想借助 MoSCoW 方法的时候再使用它。

当分解高层级用户故事（称为史诗）时，你会发现功能的集合会自动归入必须有、应该有和可以有三个类别里。

通常 MVP 可以被验证和使用，但是很难直接投放到市场中进行销售，因为他们还没有完整到能吸引用户使用。开发商业性的产品和服务需要更多的功能。

另外一个替代方案是使用最小可上市产品（MMP）。MMP 的理念基于精益中"少即是多"的原则。它仅包括满足最初用户愿意为之买单的最小功能集。通过提供产品或服务的精简版本，MMP 成为了一种缩短产品上市时间的方法。而后随着时间的推移逐步增加额外的功能，使其成为一个更完整和功能丰富的产品。

6.16 定义产品目标

产品目标是产品支持的业务目标的概括，也是产品待办事项列表中所有经过评估和排序的干系人需求的概括。只有对产品本身、客户和其他主要干系人的需求以及对产品如何支持组织目标有一个全面的了解，才能定义一个有价值的产品目标。

请注意，对于所有管理产品待办事项列表的 Scrum 团队来说，产品目标是方向（True North）的一种体现。每个冲刺目标必须与实现产品目标的过程一致并为之铺路。

产品目标并不会自行出现——因为它比需求本身还要更加的高层级，所以需要对其进行专门定制。

图 10：什么是产品目标?

图片由 EXIN 基于以下内容创建： Botha, J. (2021). *[Courseware]*. GetITright

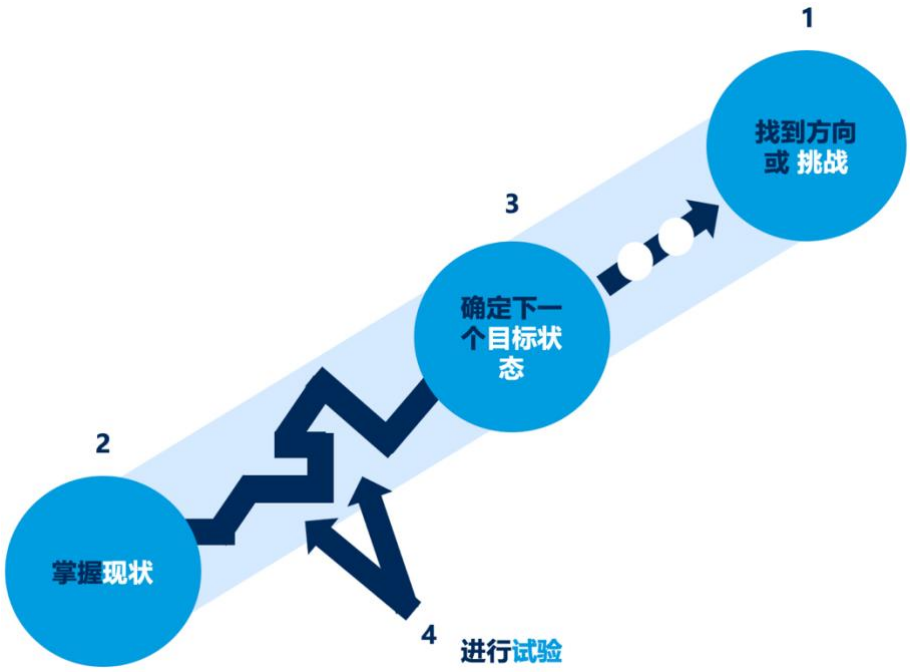
丰田改进套路（Toyota Improvement Kata）是一个比较好用的技术，它可以澄清和梳理的不只是产品目标，还包括产品待办事项列表。它所能带来的好处在开发初期尤为明显。

丰田改进套路与敏捷类似，也是一种科学的、经验性的方法，被用来理解问题、目标和目的。传统意义来说，他不是用来称其目标和目的的。然而当目的或目标没有被定义好时，问题就会出现。而此时也正是丰田改进套路大显身手的时候。

丰田改进套路一般遵循以下四个步骤：

1) **找到方向或者挑战：**理解并知晓方向和宏观的长期的愿景。
2) **掌握现状：**审查现状，能够真实并清晰地定义当前状态。
3) **确立下一个目标状态：**确定一个合格的目标去突破当前知识和能力的局限，迎接挑战并将在未来某天完成。
4) **进行实验：**有条不紊地、科学地进行实验以达到下一个目标状态。

图 11：丰田套路步骤



图片由 EXIN 基于以下内容创建：Rother , M. (2018). *The Toyota Kata Practice Guide: Practicing Scientific Thinking Skills for Superior Results in 20 Minutes a Day*. McGraw-Hill Education.

你可以使用熟悉的工具，比如故事，来制定你的目标。不过最好还是先有足够的干系人输入。你首先需要了解组织目标，以及那些推动主要的、长期的组织目标的原则和目的。

你或许可以这样开始：

- **定义组织最高目标：**在<多少年>内，<你的组织>将成为、达到、拥有<你期望的东西>。
- **为了实现设既定的目标：**我们需要<什么时候，在怎样的可衡量的标准来下完成某事>。
- **评估当前的现状**以识别要解决的差距。你可以选择逐步缩小差距，比如
  - “我们将在<何时之前><做什么>”。
  - 如果不是太野心勃勃的话，可以把同样的方法应用到既定的组织整体目标上。
- **积极地管理产品待办事项列表相**当于丰田套路中进行的实验。

我们做事的方式和顺序通常受到干系人需求和优先级的影响。在定义高层级步骤以达到目标状态的时候，我们必须考虑他们的需求。

纯粹主义者会说丰田改进套路并不是这么使用的。然而，这已经被证明是有效的做法。有很多资源和指导都适用于丰田套路，并使其成为一种可行的定义合格产品目标的方法。

和敏捷相似，丰田套路更深层次的使用也依赖于沟通、团队合作和协作。

6.17 如何收集需求

在敏捷中收集需求的主要方法是记录用户故事。

通过在陈述中定义需求的预期输出和结果，用户故事格式可以收集到不同颗粒度的需求。

一般在首次定义产品和产品待办事项列表时收集故事，在这个阶段中故事还是粗粒度的、大的故事，也称为史诗。

随着需求交付工作的开展，故事逐步被分解成粒度较小的故事，进而再分解成任务，便于在增量中处理。

在定义用户故事时，可以使用 INVEST[22]原则：

- **独立的**：每个故事都应该独立存在，与其他故事之间没有依赖关系。
- **可协商的**：故事不是合同，而是协商和变更的机会。
- **有价值的**：每个故事对用户和干系人都是有价值的。
- **可估算的**：基于相关领域和技术知识，每个故事需要的时间和预算支出都是可计算的。
- **短小的（或者简单的）**：用户故事应该足够小以易于评估和实现。
- **可测试的**：确保你可以通过故事本身解释的标准对用户故事进行测试。

人们经常发现这种需求收集和持续梳理的方式会产生一些混乱。在这种情况下，使用其他信息和文档对用户故事进行补充可能对组织有所帮助。

附加信息普遍是技术性的或者跟非功能性需求相关的，还可能包括合规或者流程标准。实际上，建议在可能的情况下，所有的技术和非功能性需求也应该以故事的形式来定义。

因为需求是不断完善的，定期地邀请干系人参与讨论甚至需求梳理活动是非常重要的。

在需求不够明确而无法采取行动时，还需要邀请其他干系人，尤其是用户一同参与讨论。干系人是需求方面的专家，他们可以分享观点、提供建议，甚至在每个迭代里帮助记录需求。

随着时间的推移持续梳理产品待办事项列表、更新需求。需求不是一成不变的，会随着时间发生改变。经常让客户和用户参与到产品待办事项列表梳理工作中，有助于识别新的需求以及变化的业务需求和优先级。

当 Developers 在冲刺中开始构建增量时，他们经常发现之前未知的信息或者依赖。这些信息也应该用于梳理产品待办事项列表和冲刺待办事项列表。

梳理冲刺待办事项列表意味着在当前冲刺中重新规划工作，但是新发现的情况通常不能在当前冲刺中完成，这时需要对产品待办事项列表进行梳理。请记住产品待办事项列表梳理是一个持续的活动，不是一劳永逸的。

---

[22] Bigelow, S. J. (2020). 7 techniques for better Agile requirements gathering. https://searchsoftwarequality.techtarget.com/tip/7-techniques-for-better-Agile-requirements-gathering

7 敏捷规划与估算

在本章中，我们将会讨论几种估算和规划的方法。但是从整体上来看，Scrum 使用了两种类型的估算和规划的技术：速率驱动和承诺驱动。两者的支持者都对他们的选择充满了热情。

速率驱动的估算和冲刺规划尝试给冲刺分配和之前的冲刺完成过的等量的工作（比如：等量的故事点和 T 恤尺码的工作）。

基于速率的冲刺估算的典型步骤是：

- **重新计算团队的速率。**当上一个冲刺结束时它可能已经有了变化，在 Scrum 的初期阶段更是如此。
- 基于产品目标来制**定冲刺目标。**
- 从产品待办事项列表的顶部开始，**选择**与团队速率一致的用户故事。
- **将用户故事分解为任务**，并确定完成每个任务所需要的时间。

当执行承诺驱动的冲刺规划时，团队必须在添加条目到冲刺待办事项列表之前做出承诺。并且一旦冲刺待办事项列表中的条目达到了冲刺可用的时间上限，团队就不再进一步地向冲刺待办事项列表中添加用户故事。

因为团队在估算里会很自然地将工作转换为所需要花费的时间，所以你经常会发现团队更喜欢用实际的天数或小时数来做估算。

冲刺计划会议通常这样进行：

- 基于产品目标来**定义冲刺目标**
- 从产品待办事项列表的顶部开始**选择产品待办事项**
- **将用户故事分解成不同的任务**，并且确定完成任务所需要的时间
- **让团队承诺**一个他们可以完成的冲刺待办事项列表，并将用户故事添加到这个列表中
- 查看冲刺时间还剩多少
- **重复**这个过程，直到无法向冲刺待办事项列表中添加更多内容为止，因为此时冲刺的时间已经被填满了

正如前面提到的，对于不同类型的估算，“采用哪种方法以及哪种工具最有效”是一个充满争议的话题。这里将介绍主要的技术，并交由团队自己来决定哪种技术最合适。

如果我们说敏捷不做预先的和详尽的计划，那为什么敏捷需要做计划？

在敏捷生命周期的初期进行规划只有一个目的——了解足够的知识，以便能够将产品添加到产品组合中去，从而有效地执行产品组合管理和产品管理。

规划工作应该要能够对未来 6 个月到一年的时间做出正确的决定，并且至少知道在该时间范围之后可能会发生什么，以及所需的投资。你需要能够分配资金和资源以实现企业产品组合所揭示的目标。

因此，预先规划应该“刚刚好”有能力对业务的优先级做出决策、对资金和资源进行分配、对业务产生最重要的积极的影响。

初期的规划总是会有缺陷，我们能从瀑布模式的项目上了解到这一点。但它也确实提供了见解，并成为整体业务的讨论、发现和战术计划的催化剂。

对于敏捷方法，特别是对使用 Scrum 的主要反对意见之一是，Scrum 团队没有高层级的视角来做出恰当的战略和战术决策。如果你以纯粹的眼光看待敏捷和 Scrum，那么这些反对意见在一定程度上是正确的。然而，目前每一种敏捷方法都有克服和纠正这些最初的反对意见的方法。

在本书后面介绍扩展时，也会提到一些当前被认为是 Scrum 的补充方法。

然而，所有的敏捷方法所遵循的原则是在任何级别上都只做刚刚好的规划，以便能够对规划或执行层面上的问题进行回答。

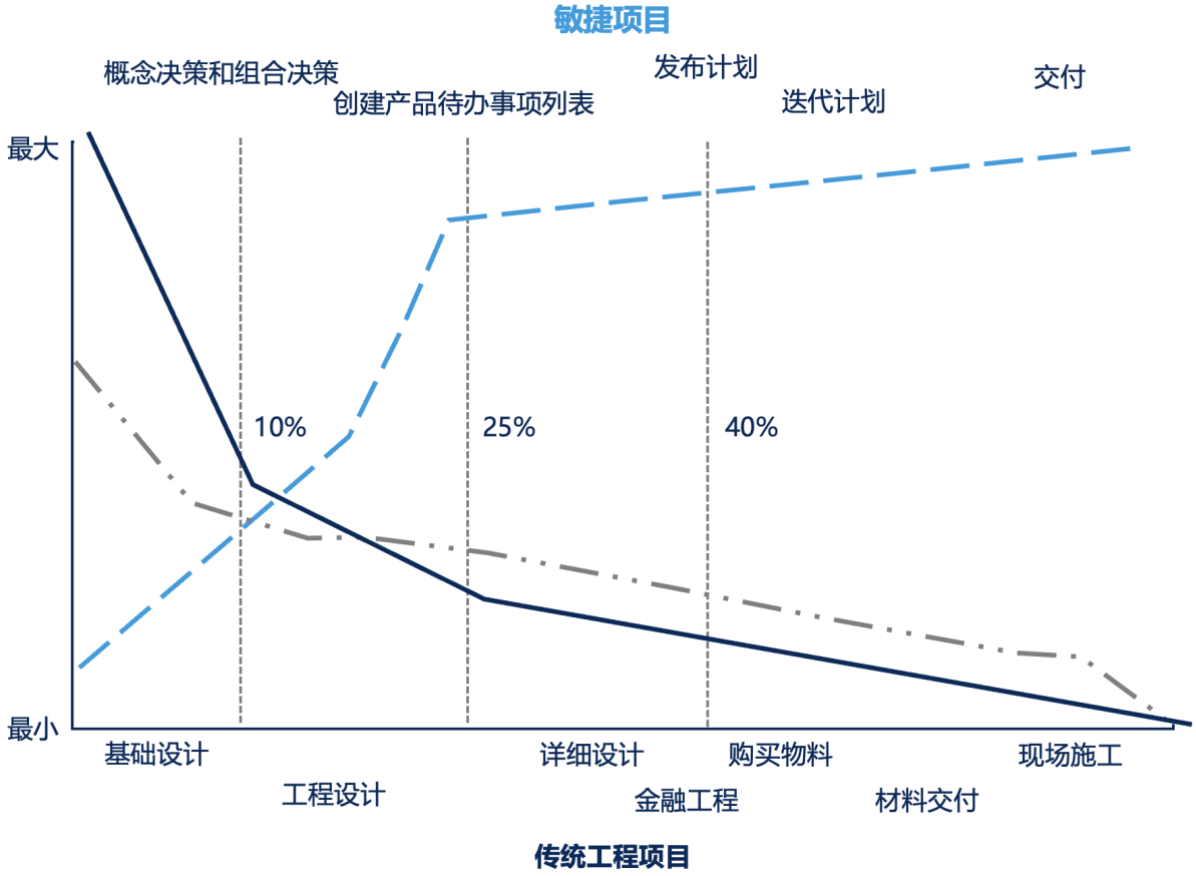
因此，假如结果证明你最初的规划和估算是错误的，那么心平气和的接受它——对瀑布式项目和敏捷项目来说都是如此。最初的规划就像是打在地上的一个桩、一个基础、一个基线。它让你可以学习和了解更多，并越来越接近向客户交付真正的价值。

Bruce Martin 在他 1980 年的一篇发表在在 PMI 的项目管理季刊中文章*《目标：提高工程进度报告的可信度》*中，绘制了一幅至今依然有用的图表。那篇文章的讨论点并非敏捷或 Scrum，而是瀑布式项目。同时那篇文章也再次证明了大多数反对 Scrum 的理由是基于一个错误的假设前提，是有瑕疵的。

那个假设具体来说就是在项目开始时，详细的瀑布式计划要比高层级的敏捷规划更好（参见下面的适配图）。

所以你可能会问，为何？因为随着时间的推移，你对投资组合越是频繁地进行高层级的估算和改进，结果就会越好。你将一个新项目和可交付成果与一个已知的、老项目进行反复比较，这也是使用敏捷估算方法的好处之一。你比较的越多，就发现的越多，也就会越做越好。

图 12：详细规划的可靠性



图片由 EXIN 基于以下内容创建：Martin, B.A. (1980). *The goal: to improve credibility in the reporting of engineering progress*. Project Management Quarterly, 11(2), 14-22.

在这个过程的早期，详细规划的可靠性是值得怀疑的。随着工作的进行，规划和估算就越可靠，对成本和进度控制的需要就越小。

渐渐的，学习和改进的过程也在使得创建和维护产品待办事项列表和路线图变得更加准确。因此，组织对未来会更有信心。

创建这个包含估算在内的所有项目的高层级视图，为项目组合管理提供了输入信息，这反过来帮助我们将工作聚焦在对组织至关重要的东西，并将针对项目的战略举措与项目关联起来。

它有助于回答以下问题——我们应该构建什么，什么时候构建，以及有必要构建它吗？

规划也有助于指导子项目，如市场营销、产品品牌管理、销售和广告，并经常启动培训和技能提升计划，为交付项目做准备。

一个好的计划拥有足够的细节，可以在你想要使用计划的层级上进行规划工作。这意味着敏捷规划是迭代性的，且具有明显的规划期限限制。在这个期限内，所有规划的内容都是需要尽快处理的。

7.1 是什么让敏捷规划与众不同?

实践者应当小心谨慎，不要用传统的方式进行规划。传统的项目计划是线性的和基于活动的。

在敏捷中，价值不是以线性的方式交付的，而是通过不断调整功能的优先级来为客户释放出最大的价值。

除了线性计划和执行的缺点之外，每当延迟影响到关键路径时就会有各种应对方法出现，瀑布方法总认为能完全预料未来。然而这是不切实际的幻想。

因此，在瀑布式项目中，在关键路径上的每一次延迟交付都意味着项目向客户交付价值的延迟。

另一个问题是，根据帕金森定律，项目团队会自行填补由延迟导致的"空闲时间"或"等待时间"。出现这种现象的原因可能是因为在此类性质的项目中的主要衡量标准，就是你至少目前手头上得有事做！下游的空闲时间也不会用于非依赖性的任务，因为它们不是下一件要做的事。因此，延迟的影响几乎总是被传递给下游。

记住敏捷团队的四项价值观：

价值观一：**个体和互动** 高于 *流程和工具*。

价值观二：**工作的软件（或其他任何您的项目需要交付的东西）** 高于 *详尽的文档*。

价值观三：**客户合作** 高于 *合同谈判*。

价值观四：**响应变化** 高于 *遵循计划*。

这些价值观应当浸入你所做的每一件事，包括你的计划方法。

请注意，敏捷规划应当在你项目工作开始之前占有一席之地。这是规划的最佳时机。因为从创建最初的计划和产品待办事项列表开始，我们已经积累了很多经验。敏捷和 Scrum 的迭代性质意味着，所有的经验教训都可以帮助我们提出正确的问题并找到可能的最佳的方案。

以团队的形式工作。不同的团队在不同的层面上工作，正如上面的规划范围所强调的那样。同时在规划的各个层次上都要包含广泛的利益干系人。对话、询问、探索并发现最能代表各方的智慧和理解的东西。

不要试图去定义那些未知的东西，将它们留到日后再说。随着工作在每个迭代逐步完成，迭代计划都会涌现出来并变得更加详细。

计划应该是轻量级的，这与精益的原则"刚刚好"便能够做出好的决策如出一辙。不用更多，不用完美，也不用追求完美，只要刚刚好就行了。

在做规划时，建议使用参与式技术与可视化技术。如果墙上没有贴满便利贴，板上没有涂鸦满，那表明你可能做的还不够。

此外，确保规划能恰当的传递到下一层的参与者。不要未经磋商就将规划丢给其他人，那通常都不会有好的结果产生。如果下一个活动需要的伙伴不在房间里，那就去找到他们进来，解释、讨论、辩论并达成共识。

敏捷规划是基于功能创造的价值，并且基于价值来分配优先级。详细的规划不是在敏捷的初期完成的；在高层次的敏捷规划中甚至没有具体活动。

这种方法有四个显著的好处：

1. **没有基于活动的规划**——因此避免了该方法的缺点。
2. 完工的本质是交给专家去执行价值的交付。目前为止，他们是决定最佳方法的最佳人选。
3. 因为不用担心那些活动，所以可以**站在更高的层面进行规划**，考虑为客户和组织创造价值的最佳顺序。
4. 因为没有所有活动的详细视图，所以你可以**接受自己对未来有一个不完美的想法**。在走向未来的旅途中，不断尝试更好地理解需要做什么，并努力减少不确定性。

上述最后一点也意味着需要同干系人沟通，让他们知道项目组合或者路线图并不是一种承诺而是计划进度的展示。当组织或客户环境或优先级发生变化时，一切都可能随之发生变化。

你也许会问，这种方法是否会给客户带来很多不安？毕竟客户需要的是确定性。

答案是肯定的。至少：一开始是肯定的。

为什么？

因为客户必须在整个价值交付周期中持续参与规划，而 PO 的工作就是保证让他们参与进来。

前面提到过 Scrum 团队参与产品待办事项列表的梳理和优先级的调整。作为业务代表的 PO，在梳理产品待办事项列表时也需要与业务同行就业务优先级进行对话。

当讨论 Scrum 团队的任务时，我们曾经说过他们需要计划迭代或冲刺中的工作。我们还指出，PO 需要在产品待办事项列表（产品级别）中做计划和排序，以便最好地计划来执行发布和迭代。如果客户有任何优先级变更，客户需要向产品负责人同步但产品负责人需要确保不能随意变更客户的优先级。

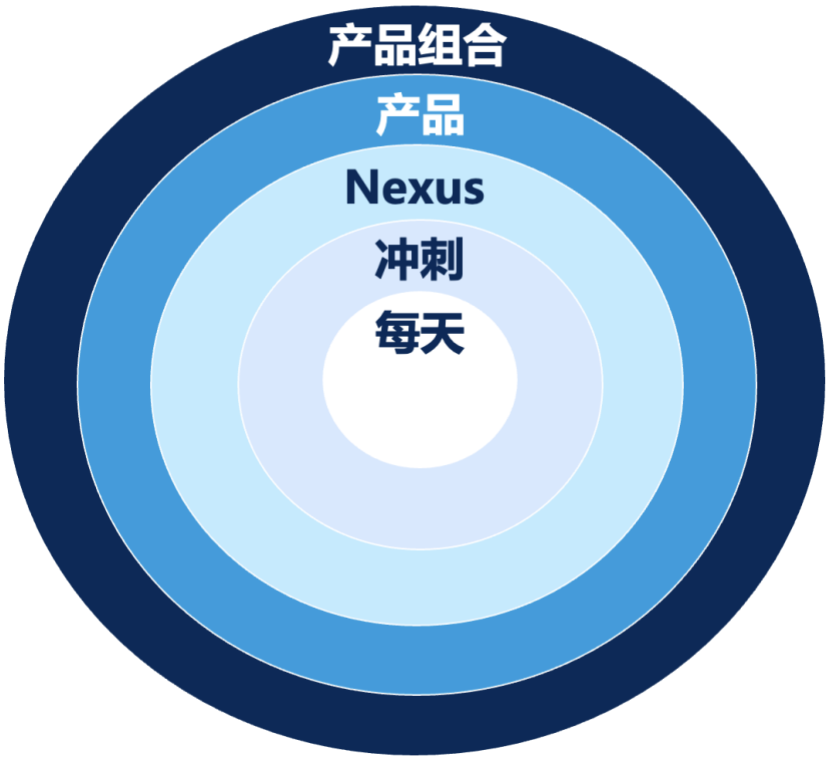
若要将项目纳入组织的投资组合中，则需要对完成项目所需的总工作量、预算和资源进行估算。这些信息也被用来对该项目分配优先级。这些信息需要在组织内其他项目的情境中被查看，从而确保优先级分配的正确性。

这代表了至少需要四个层级的估算和规划来贯穿项目生命周期。在许多有着复杂项目的组织中，甚至会有多达四、五或八层的规划和估算。这当中的关键是，每个层级所对应的估算和规划都只需要回答必须的问题。

细化规划和估算的级别通常是：如果要把一个项目加入进投资组合，或者创建与规划产品待办事项列，或者规划一个大规模的开发或发布、或者计划和执行冲刺，我们需要知道些什么信息来帮助我们做决定？

规划和估算也会在冲刺中进行：我们在这次冲刺中要做什么，我们今天要做什么？

图 13：随着我们越来越接近工作的完成，计划的层数也越来越详细。敏捷在每一层都使用"刚刚好"的计划原则



图片由 EXIN 根据以下内容制作：Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value*. Amazon Digital Services LLC – Kdp Print Us.

之前讨论过针对产品、发布和冲刺级别的规划和估算，那么对于项目组合的规划和估算呢?

为了回答这个问题，就需要知道在创建和更新组织的项目组合时需要询问哪些问题。

组织组合的目的是管理组织目标和目的的实现。问题的本质是：需要什么项目才能实现目标和目的？然后对确定的项目进行评估和排序。

- 那我们首先需要做什么?
- 什么又会产生最重大的影响?

不幸的是，这个问题并不那么简单，因为你总是面临着各种约束，通常是资源和预算方面的约束。

然后问题就变成了：

- 那些需要做的事情该如何整合在一起才会产生最重大的影响?

为了回答这个问题，需要对项目所创造的价值有很好的了解，比如完成它需要多久，以及成本和使用的资源等等。其次，还需要考虑相互依赖关系。

可以理解的是，这个问题的答案充其量只是一个有根据的猜测，但是与之前的项目和已知的工作相比，这意味着该水平的估算可以很快达到 80%以上的水平。对于项目组合规划而言，80%的估算就足够了!

你是否注意到回答这些问题不仅仅涉及到待完成的工作的性质。所涉及团队的速率也会产生影响！

如果不知道这项工作最终由谁来完成，那么还是建议按照所有团队的平均速率来工作。

7.2 谁参与规划?

非常好的问题，答案取决于所涉及的规划层级。

对于项目组合计划，人们希望 PO（产品经理）和他们的业务发起人（客户）以及高级经理或执行经理一起参与讨论。

产品待办事项列表的创建者应当包含 PO、业务发起人（如果有的话），以及早期的各类用户代表们。使用者需要对产品待办事项列表中的故事以及他们的相关顺序有一定的了解。

顺序的选择是经深思熟虑的；产品待办事项列表不仅仅使用重要性和优先级来决定工作的顺序。其他因素，如风险、依赖性等等也可能对排序产生一定影响。

理论上，产品待办事项列表梳理有着相同的参与者。但现实情况是，各类用户代表很可能不是必要的。除了这些角色之外，还将目前正工作在产品待办事项列表上的 Developers 和 SM 们包含进来。Scrum 团队所提供的学习到的经验也是至关重要的。

类似的参与者也适用在规模化实施和冲刺规划。随着规划的进度发展到特定的冲刺，客户和用户的参与变得越来越少。但这并不意味着他们退出了。在整个冲刺中，用户和客户的参与度较高的 Scrum 团队要比那些没有参与的团队做的更好。所以并非得等到冲刺评审才获得用户和客户的反馈。

PO 可能需要邀请多个团队的 Developers 和 SM 一起参加对产品待办事项列表的定义、梳理和规划活动。

规划越接近实际的冲刺，技术性就越强。到了那个阶段，需要了解不同的团队如何在同一个产品待办事项列表中工作、它们的依赖关系和结果、如何排序工作和活动、需要什么样的沟通，以及最终在冲刺中各自的工作任务。这就是为什么在更大、更复杂的环境中充满凝聚力的方法如此重要。

尽管将所有类型的计划都看作具有时间盒限制并不是一个好主意，但它通常还是作为具有时间盒限制的事件被执行。

规模化实施和冲刺都应该有目标、目的，以及对于交付完成的成功或验收的标准。但是，在规模化实施层面，它应该更灵活一些。然而每个冲刺都会有一个明确定义的冲刺目标以及每个增量的 DoD，其中可能也会包含进一步的、详细的验收标准。

7.3 估算技术

在做规划时需要进行估算是显而易见的事情，但这句话却透露给我们非常重要的信息。

估算很难做到完美，其诀窍在于知道估算只需要做到哪种程度即可。

需要考虑的两个因素是**准确性**和**估算所需工作量**。如同生活中绝大多数事情一样，如果将不同的尝试结果绘制成图，和可能会显示出一个典型的钟形曲线。获得更好估算的一种方法是获得更有意义的数据，这样团队做得越多，自然就会做得越好。

越多的团队成员积极参与并获得估算经验，团队将能更好地完成高质量的估算。你会发现，所有敏捷估算技术都依靠团队共享他们的信息。

假设一个故事看起来很难估算。在这种情况下，团队应当考虑将故事划分成更容易估算的小故事，然后再将小故事的估算总和作为对大故事的估算。这种技术被称为“分解-聚合”。

但是如果团队不清楚如何下手怎么办呢?

你可以选择向一位或多位专家请教。然而请注意，这并不会使估算更精准，因为团队之外的专家们并不了解团队的技能和能力。但无论如何，这样肯定比猜想来得靠谱。

团队还可以尝试将当前的工作与之前完成的工作进行对比。实际上我们就是这样来使用理想人天或故事点。但是如果没有可以比较的项目，那唯一的选择就是向外界寻求意见。

7.3.1 重新估算

当使用故事点和理想人天时，意味着一旦工作分解发生，就需要重新估算。这些技术用于简化对新功能或故事及其所有相关的复杂性的估算，它不必对其包含的任务进行工作分解。

由此可见，一旦冲刺规划中发生了任务分解，就需要重新估算，以便可以马上将相关联的任务的估算总数合计并入功能或故事中去。

你可能会认为这是多余的，因为你不再关心这个级别的故事或功能了。然而，聚合被证明是有意义的，因为团队可以从最初估算和现在的更详细的估算的差别中吸取教训，这将显著的为团队未来进行类似性质的工作能力带来好处。

回顾性讨论的主题之一可能是发现过去的估计、详细的估计和完成工作的实际时间之间存在重大差距的情况。

考虑到团队很可能会在冲刺完成后立即帮助产品负责人改进产品待办工作事项，因此建议将其作为议程项目。

因此，包括该议程项目将立即有助于改进未来的估算。

另外需要记住的是，部分完成的故事需要在冲刺结束之后立即添加回产品待办事项列表中去。这些故事需要被重新估算，因为它虽然有部分工作已经完成，但它依然存在部分未完成的工作。

在冲刺规划章节中，我们将会介绍一些估算技术，这些技术在估算产品待办事项列表条目时也能用得上。

7.4 在冲刺计划会中还需要做些什么?

再次提醒，在冲刺计划会上需要回答以下问题：

- 为什么这个冲刺是有价值的?
- 在这个冲刺可以完成哪些工作?
- 这些工作将如何完成?

在计划活动结束时，需要清楚定义下列各项：

- 冲刺目标，
- 为冲刺挑选的产品待办事项列表条目，
- 以及如何交付冲刺待办事项列表的计划。

之前已经概述过了冲刺计划，但是本节仍然扩展了一些可以用在冲刺计划会上的技术。

7.4.1 冲刺计划

冲刺计划是一个具有时间盒属性的事件，冲刺中每周大约需要 1-2 个小时的计划时间。

在冲刺计划中，SM 会指导 Developers 对功能进行估算，但 SM 不会决定估算结果。同时 Scrum 团队约定完成一组来自产品待办事项列表条目，来使组织离实现产品目标更进一步。这样的约定通过冲刺目标定义和为冲刺待办事项列表添加条目的形式而体现出来。

那如何知道需要承诺多少工作?

在开始阶段，必须对待办事项列表里的项的估算时间进行实验，直到有一些可以体现人员工作速度的数据。这被称为团队速率，稍后会对此进行详细的介绍。现在，让我们把注意力集中在被称为"冲刺计划"的具有时间盒约束的事件上。

冲刺的准备工作中冲刺计划之前就已开始了。产品负责人需要确保产品待办事项列表得到恰当的梳理、分解，并且其中条目已经被尽可能地排序和估算大小，以便在冲刺计划会议期间进行富有成效的讨论。

整个 Scrum 团队都要参与到冲刺计划中来。这样做的目的是想要在产品负责人和其他团队成员之间就冲刺中应该包含哪些工作达成一致，这当中也包含了对冲刺目标达成一致的要求。

Scrum 实践者通常把冲刺和增量这个词当作一回事，但根据 Scrum 指南，冲刺可以包含若干个增量。增量被看作是实现产品目标的坚实的基石。

每个**增量**都附加到所有之前的增量上，并被彻底地验证，以保证所有的增量都能一起工作。由于要提供价值，所以增量必须是可用的。

即便是在同一个冲刺中也能创造出多个增量来。

全部的增量会在评审会上展示，从而支持经验主义[23]。然而请注意，增量可以在冲刺结束前就交付给干系人。不应把冲刺评审会当作交付价值的唯一出口。

---

[23] 这里说的大概是通过反馈来获得调整的机会，而不是通过严格的管理学过程来获得反馈。——译者注

工作本身不能当作增量的一部分，除非它符合 DoD。

这个定义意味着每个增量都需要定义"完成"是什么，而冲刺目标则是整体可交付的成果。

冲刺计划从一场围绕理想和广义的冲刺成果的对话开始。产品负责人将向团队概述产品待办事项列表顶部的条目，并且提醒他们为产品目标做好相应的准备。

这为冲刺定义一个初始和广义的目标树立了背景。广义这个术语是经过深思熟虑的，因为如果过于狭隘，那么团队将无法选择他们在冲刺中要做什么不做什么——这个与自管理团队的敏捷原则，即自主决定做什么以及如何做相矛盾。

这个对话的结果是为冲刺定义了它的目标，并且冲刺目标应当支持整个产品的目标。

冲刺目标是为特定冲刺设定的目标，团队可以通过实现产品待办事项列表中列出的条目来满足该目标。冲刺目标是产品负责人和 Developers 协商的结果，并且是具体的、可度量的。Developers 需要确保目标是现实的，因为这是他们承诺完成的目标。

为了确定冲刺目标，Roman Pichler[24]提出了三个需要考虑的问题：

- **为什么要实行冲刺？**为什么运行冲刺是值得的？应该实现些什么？
- **如何达到目标？**使用了哪些工件、验证技术和测试组？
- **怎么知道目标已经实现了呢？**例如，至少五分之三的用户在不到一分钟的时间内成功完成了可用性测试。

下面几个问题的答案同样重要：

- **为什么这个冲刺是有价值的？**PO 提出该产品如何在当前的冲刺中提高价值和效用。然后，整个 Scrum 团队共同定义一个冲刺目标，以传达为什么冲刺对于干系人是有价值的。冲刺目标必须在冲刺计划会结束之前确定下来。
- **在这个冲刺中可以做些什么？**通过与 PO 的讨论，Developers 从产品待办事项列表中选择要包含进当前冲刺的条目。Scrum 团队可以在这个过程中梳理这些条目，从而提升理解和信心。选择在冲刺中可以完成多少任务可能具有挑战性。但无论如何，Developers 对他们过去的表现、即将到来的能力以及他们对 DoD 了解得越多，那么他们对冲刺的预测就越有信心。
- **所选中的工作将如何完成？**对于每个选定的产品待办事项列表条目，Developers 计划了必要的工作，以创建满足 DoD 的增量。这通常是通过将产品待办事项列表条目分解为一天或更短小的工作项目来实现。如何做到这一点完全由 Developers 自行决定。没有其他人告诉他们如何将产品待办事项列表的列表项转化为价值增量。

在这两组问题之间，可以创建一个精心设计的冲刺目标。从那刻开始，Scrum 团队所作的一切努力都将通过它对冲刺目标的贡献来衡量。

下一步是评估团队是否能够在定义的冲刺时间范围内切实地完成所需的工作，从而达成冲刺目标。通常，关于这个冲刺的讨论将开始强调在当前冲刺之后的未来的冲刺中可能采取的后续步骤。

---

[24] Overeem, B. (2016). The 11 Advantages of Using a Sprint Goal.
https://www.scrum.org/resources/blog/11-advantages-using-sprint-goal

这意味着 PO 可能不得不重新排序产品待办事项列表条目，因为冲刺待办事项列表条目应该反映产品待办事项列表顶部的部分，而下一个冲刺可能的待办事项一般来说就是产品待办事项列表中接下来的部分。这也有助于最大幅度减少为下一冲刺阶段选择条目的时间，因为那时来自 Developers 的信息已经能反映在产品待办事项列表中了。

冲刺中包含的产品待办事项列表条目数量取决于团队的能力和速率。刚开始使用 Scrum 时只能是猜测，因为团队没有可用的数据来计算团队的速率。

任何选择纳入冲刺的条目都必须已经被分解成小的、可估算的和可测试的条目。而且它们应当在团队将这些条目添加到冲刺待办事项列表之前进行分解。

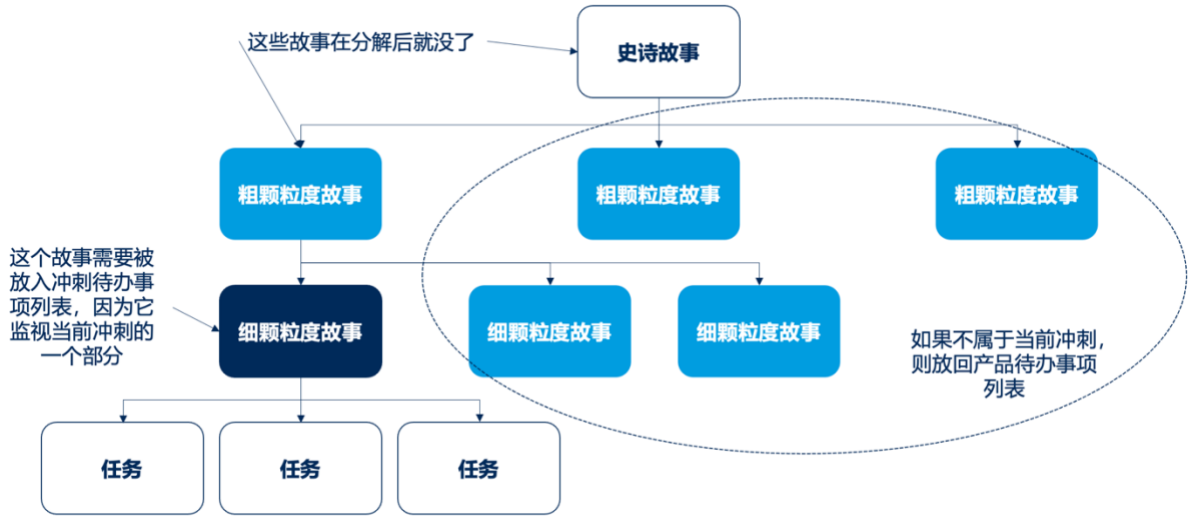
条目在添加到冲刺待办事项列表之前将被重新估算，因为不同的开发团队可能在技能、能力和速率上处于不同的级别。

尽量让条目尽可能小，绝对不要让其持续太长时间。这将有助于将工作分解为简单和容易处理的任务。同时也要对在制品（WiP）进行了限制，并使燃尽图/燃起图让进度追踪更加简单。

请注意，用户故事可以在多个冲刺中被分解多次。在当前冲刺中，没有必要分解更高层次故事的所有内容，除非它们构成了冲刺中要完成的工作的一部分。我们要做的只是分解要在这个冲刺中完成的工作。

如果团队在这个冲刺中没有处理分解后的故事，那么可以按适当的优先级和估算将它放回产品待办事项列表。

图 14：从史诗故事到任务——将需求进行拆解



图片由 EXIN 基于以下内容创建：Botha, J. (2018). *Scrum Masters and Product Owners [Courseware]*. GetITright.

同样，考虑在冲刺期间可能影响当前项目情景下的绩效的问题。强调相关性并考虑其带来的影响。

通常情况下，会发生对于非 Developers 的依赖。尽管最好能够不惜一切代价避免这种情况，但它还是会发生。所以要考虑如何消除依赖关系，如果不行，那就考虑如何最好的管理它们。

有一个案例就是，Developers 需要在云环境中工作。提出需求可能很简单，但是无法预测何时满足需求。能做些什么来确保它不会影响冲刺的绩效吗？如果有的话，那应该做些什么，而你能否采取恰当的步骤来实现？

一旦团队就冲刺中要纳入的条目达成一致，团队就应该制定一个冲刺待办事项列表条目，在其中包含要交付的条目。冲刺待办事项列表要定期更新；建议每天更新一次，尽可能保持最新状态。

然后，团队需要就构成冲刺的每个增量的 DoD 达成一致，这将是团队对增量和冲刺中需要完成的条目的工作的承诺。

DoD 是使增量成功所必须满足的需求列表。一些开发团队在他们的 DoD 中包含了需要完成所有待办事项列表条目，而有些团队则没有。

DoD 为每个人提供了就“哪些工作应该作为增量的一部分完成”的共同的理解。如果一个已经完成了的产品待办事项列表条目不符合 DoD，那么它就不能在评审会中使用或呈现。它必须被退回到产品待办事项列表中，以便加入到将来的冲刺中去。

如果增量的 DoD 包括了组织标准的一部分，那么所有 Scrum 团队都必须最低限度地遵循它，例如，将非功能性需求定义成安全标准或质量需求。如果需求不是一个组织标准，那么 Scrum 团队必须为产品和产品需求创建一个合适的 DoD。

Developers 的可交付成果必须符合 DoD，如果多个 Scrum 团队合作同一个增量，他们必须相互定义并遵守相同的 DoD。

当产品待办事项列表中的待办项满足 DoD 时，增量就诞生了。

注：使用 MoSCoW 方法的团队经常从“必须有”、“应该有”和“可能有”中选择待办项包含进冲刺。通常，80%是“必须有”（要完成的高重要性的条目）和“应该有”（有用的、有价值的但不是必不可少的）然后 20%就是“可能有”。通过遵循这种方法，一旦发生了不可预见的事情，也可在冲刺中创建一个缓冲区，从而提升冲刺的成功率。

然而，在团队含有缓冲项（应该有和可能有）的场景中，通常只有“必须有”的故事被包含在 DoD 中。

Scrum 中的标准实践是在冲刺待办事项列表中只包含产品待办事项列表顶部的条目。然而，许多 Scrum 团队已经采用了 DSDM 实践，如果这种方法适合您的组织，那么就放心大胆的使用它就好。

在选择要包含在冲刺待办事项列表中的条目时，考虑依赖和相互依赖关系是至关重要的。依赖关系通常是技术或基础设施性质的。如果存在依赖关系，也要记得把它所依赖的故事包含进冲刺中。

故事什么时候完成?

当它满足增量的 DoD 时。

故事和相关任务也应该有验收标准。它们是在冲刺开始时将故事分解为任务时设置的。活动和相关的标准可能会随着开发的进展和发现更多关于特性/需求的内容而变化，除此以外，所有的工作在确认可用之前都必须根据验收标准进行测试。

上面的描述听起来非常像软件开发中使用的极限编程的“测试驱动开发”方法。在这种情况下，“完成”的最终测试是询问软件是否可部署、可用和安全。

*完成*不仅仅是满足个人设定的标准，也意味着团队中增量中完成的内容已经准备好使用并且可用。

注：如果正在完成的工作是软件的开发，那么完成意味着所交付的内容可以进入持续集成/持续交付（CI/CD）流水线，并且增量几乎立即可用。

CI/CD 是使用预定义的工具和自动化工作流，以尽可能快地将软件代码交付到实际环境时所使用的概念。用于以尽可能自动化的方式对软件进行测试和正确部署的首要准则，通常也被称为 DevOps。尽早地将 DevOps 和 Scrum 整合，是一种为用户和客户创造价值的好方法。

一旦团队就冲刺待办事项列表中应该包含的内容达成一致，冲刺计划的其余时间就应该用来对每个用户故事进行工作分解、任务估算和工作项目的验收。

在冲刺计划的后部分中，PO 的角色是澄清问题和详细阐述验收标准。PO 在评估团队如何完成他们的工作方面没有作用。此时，他们所能提供的帮助，仅限于对待办事项列表的相对优先级排序。

SM 需要确保在会议中提出的新问题和担忧，或者在计划中被识别的假设、发现的依赖关系，都尽可能早地被记录下来并被更好地理解。这些工作宜早不宜迟。

7.4.2 为条目估算大小

在之前的创建产品待办事项列表和梳理章节中引用了估算技术。类似的技术也用于分解冲刺和产品待办事项列表的故事。

尽管在梳理产品待办事项列表是对条目进行过估算和分配大小，但是仍然需要根据目前已知的情况重新评估冲刺待办事项列表中的内容。在重新评估条目时，需要考虑需求、环境以及 Developers 的能力和产能的各种变化。

根据实际情况适当为条目分配大小非常重要；否则，团队可能无法在冲刺结束时按照 DoD 交付。

通常，冲刺待办事项列表包含着一些条目，但当进行任务分解时，发现估算的时间和工作量是不匹配的。团队可以重新审视冲刺待办事项列表的内容，并与产品负责人协商，如果一些待办项不能在冲刺期间真正的完成，就把它们放回到产品待办事项列表中。剩下的工作余量可以通过在冲刺待办事项列表里放入其他较小的产品待办事项列表条目来填补。

要记住，当这样做时，冲刺目标和 DoD 可能会受到影响而需要修改。最好只做那些不影响冲刺目标的变更，但由于你本质上还在忙着制定计划，所以不可避免地会需要修改冲刺目标。

一旦冲刺计划完成了，就不要做出会影响冲刺目标的变更。下面将简要描述多种为条目分配大小的方法。

7.4.3 故事点

尽管故事点没有正式的定义，但通常来说，故事点是一个相对度量单位，把将要执行的工作与过去完成的类似工作进行比较。

因此，故事点是相对的，需要与 Scrum 团队的速率一起考虑。这种估算方法在应对工作环境中的变化时更加稳健。

你可能会说，与之前完成的工作相比，新的工作将可能只需要一半或者三倍的时间。但是，用来比较的单位保持不变。因此最好将参考单位选择为中尺度或中低尺度，而不是最小的尺度。举个例子，如果你用 1 到 10 的刻度，参考单位最好是 3 或

4。尽管通常的刻度比例是 1 到 10，你也可能时不时会有一个故事有 20 个故事点；这只表明它是一个需要进一步分解的史诗故事。

在产品待办事项列表级别，PO 更容易使用这种技术，因为他们通常不太了解其他技术，比如小时数。

故事点的变体包含一种叫 T 恤尺寸的用法。你可以用下面的表格作为典型的 1 到 10 个故事点的参考：

表格 1 – T-恤尺寸与故事点进行比较

T 恤尺寸	故事点数量
XS	1 故事点
S	2 故事点
M	3 故事点
L	5 故事点
XL	8 故事点
XXL	13 故事点
XXXL	21 故事点

请注意，这里使用的是斐波那契数列作为故事点的相对参考。

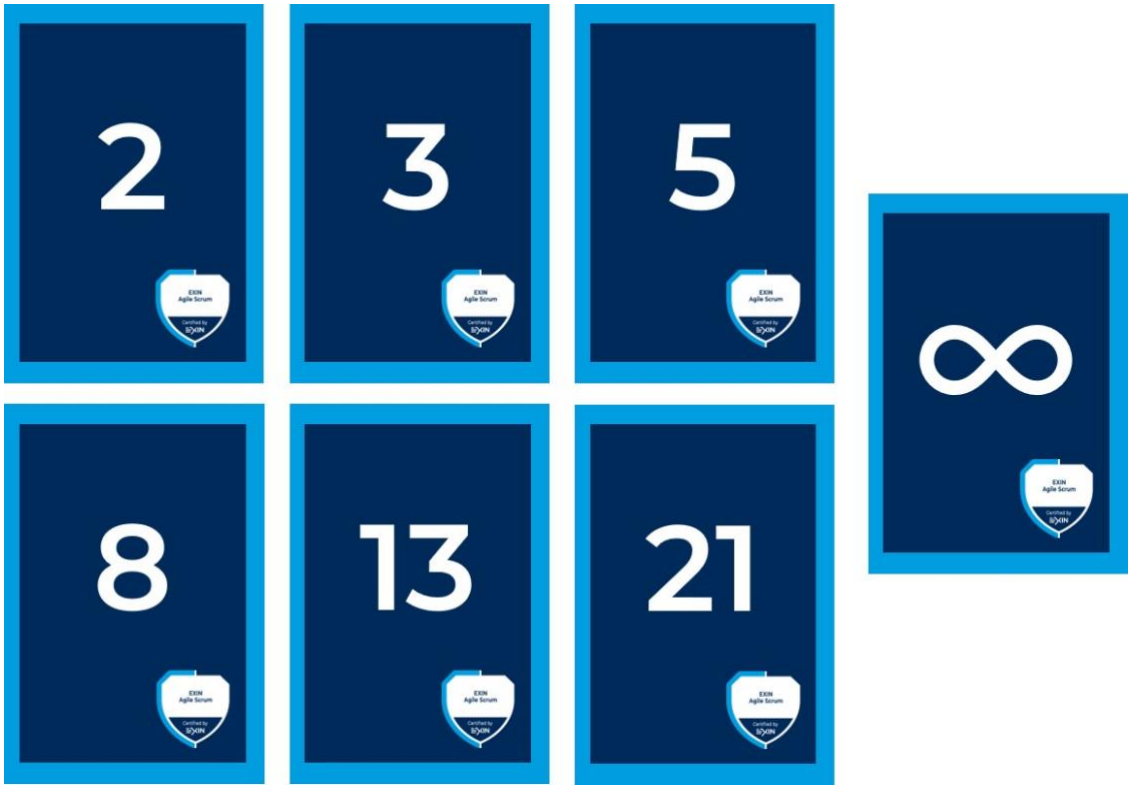
7.4.4 计划扑克或 Scrum 扑克

计划扑克实际上更像是在玩“优诺”（Uno）或“捉对儿”（Snap），而不是普通扑克。

每个 Developers 都会得到一副斐波纳契数列的牌。当被问及工作大小的估算时，每个成员都放一张卡片在桌上，卡片上的数值要么与他们的估算相同，要么是下一个更大的数字。

SM 和 PO 不应该参与估算，因为估算是从开发团队那里获取信息。PO 会在向产品待办事项列表添加条目时进行估算。开发团队的估算过程也是 PO 从做实际工作的人们那里获得更好见解和信息的一个机会。

图 15：估算扑克



图片由 EXIN 基于以下信息创建：Botha, J. (2021). *[Courseware]*. GetITright.

这里的目标是衡量团队成员之间关于完成活动或故事所需要付出的努力的分歧程度。

如果估算结果分布比较集中，那就相当容易达成一致。为了谨慎稳妥通常就是选择更大的那个数字。

然而，真正的价值在于那些异常值。例如，如果团队中的大多数人说 3 或 5 个故事点，但是这个时候有一个人说要 20 个故事点，那么就存在两种可能性。要么是出异常值（20 个故事点）的人不了解要做的工作的范围和情景，要么是他知道一些团队其他成员所不知道的东西。

对于异常值，总是会这样问：你的理由是什么？

解决第一个问题很简单；通过解释上下文，这名参与者可能会决定在下一轮出牌中修改自己的估算，如果是这样的话，就进行另一轮。

第二种可能性经常会导致非常有趣的讨论和新的见解，而这是 PO 和其他团队成员之前都不知道或不理解的。这种对话通常会导致新的依赖关系和风险被挖掘出来，并且经常会产生额外的故事。这些故事也应该被添加到产品待办事项列表中去。

继续这个过程，直到桌上有一组相对集中的值。

正如之前提到的，如果参与估算的有 10 人，其中 8 人估算任务是 3，2 人估算任务是 5，那么就选 3。如果一半的人选择 3，一半的人选择 5，那么就选择 5。不需要浪费时间让每个人就同样的数字达成一致；只要估算是相似的，那就是一个可接受的结果。

7.4.5 理想人天或理想人时

理想人天或人时的估算使用真实时间作为所需工作量的指标。要想达到一个理想的人天，需要满足以下条件：

- 当你开始工作时，手上已经有了所需的一切。
- 你唯一需要做的就是这项已经估算后的故事。
- 工作过程中你不会被打断。

再次强调，估算是**整个团队**对这项工作的估算。因为并不是所有的团队都具备相同的能力、容量或速率。

顺便说一句，扑克上的数字可以很容易的代表人天或人时，所以 Scrum 扑克的用法也可以用于理想的人天和人时。

7.4.5.1 但为什么是理想的人天或人时呢?

这里的估算是基于团队在没有被中断的情况下完成一个故事所需要的人天/人时。所以它代表了一种理想情况，没有干扰、打断也没有任务切换。这类干扰通常每天占用 2 小时，这意味着理想的一天平均有 6 小时。

现实情况是，即使使用故事点数，在某些阶段还是需要映射到时间数。有些人可能会说，如果团队的速率是通过故事点来衡量的，那么就不需要这么做了。这种说法理论上正确，但忽视了实际应用的环境。即使一个团队可以在一个冲刺中完成 60 个故事点，但冲刺是一个具有时间盒限制的活动，所以实际上无论如何都是映射到时间数上。

使用故事点的真正价值，是改变组织中的文化，至少在开始时是这样。在大多数组织中，任何给出的时间表都被默认视为一种承诺。使用通过使用故事点，可以避免有人说：“但你说这只需要 8 小时就能完成这项工作”。

当和客户交谈时，千万不要使用理想人天或人时。因为他们会把这看作是一种承诺。

7.4.6 用便利贴快速估算

如果团队的时间紧迫，需要使用某种其它的估算方法，那么可以使用一种被称为罗曼-皮克勒（Roman Pichler）的方法。

把斐波那契数列写作一块白板上，数字间留下一定的空间，给每个团队成员几个工作条目进行估算，让他们将工作条目写在便利贴上，并其贴在认为合适数字下面。

一旦所有人都做完了，让所有团队成员查看白板，并找出他们认为可能有问题或者没有正确估算的条目。允许成员立即移动便利贴，或者修改估算。

另一种选择是，在将便利贴移动到白板上合适的位置之前团队先进行一次快速的对话。

嘱咐团队成员不要对估算吹毛求疵，因为这可能导致每一个便利贴要么被转移，要么不停辩论。这可不是我们的目的。

请理解，这种方法可能不那么准确，但是它可以让团队快速地完成许多工作。

你很可能会发现，如果从一开始就使用这种方法，随着团队的成熟，它会逐渐产生更好的结果。

7.4.7 其他须知

无论使用哪种方法来估算故事或任务，在一开始，你都很可能会错。不过别担心，你做的越多，你的估算就会变得越好。

有些人说，只有那些对任务或故事非常了解的人才可以进行估算，但是让整个团队都参与进来往往会产生新的见解。其次，这也能帮助团队更好的理解工作条目，并最终有助于学习和成长。

如果一个团队成员真的觉得他们对涉及到的工作量毫无头绪，那么允许他们放弃当前估算。不必每轮都给出你的估算观点。

7.4.7.1 但是如果你不知道如何估算工作量该怎么办?

很简单，你需要发现更多信息。这样的情况经常出现在当团队必须处理一种新工作类型时。

新发现可以变成一个需要优先完成的新故事，只有将它完成，才能继续完成特定的工作条目。往产品待办事项列表里添加一个故事来找出你所需要的信息。

这通常是通过创建某种模型或原型，以及通过做研究和与可能有一些经验的其他人沟通来实现的。

7.4.7.2 那么非功能性需求呢?

什么是非功能性需求？人们通常将这些需求视为非用户需求或技术/基础设施需求。但这不是它们在 Scrum 中的定义。

非功能性需求是与功能无关的系统级或产品级的约束，是指诸如性能、准确性、可移植性、可重用性、可维护性、互操作性、容量等属性。非功能性需求是干系人完全接受产品待办事项列表条目的必要条件。除个别情况外，非功能性需求通常与质量标准相关。

许多时候，功能性需求的接受程度取决于非功能性需求。在估算功能性需求时，还应该考虑其所依赖的非功能性需求。

当把一个非功能性需求放到 Scrum 面板或看板面板上时，就好像你必须一直把它放到板上一样。因此，当你发现自己有一个需求因为别人的依赖而不断的出现时，那它就很可能就是一个非功能性需求。

你不得不在每个冲刺中包含一些非功能性需求，因为一旦这些需求没被完成，产品或服务就无法使用或者存在缺陷。所以，确保相关的非功能性需求也被包含在 DoD 中就很重要了。

7.4.8 为其他干系人写的用户故事依旧是用户故事

技术需求，即用户或客户未作要求，但是为了顺利工作，仍然必须落实到位的其他需求，算是一种不同类型的用户故事。在这里，用户是内部员工，对用户和客户的产出起到促进作用。

在精益中，这些需求可能被认为是无增值的，但却是必要的，因为客户没有要求过它们，而且很可能还不愿意为它们买单。但如果忽略它们，又将永远无法满足客户的需求。

有些人喜欢将这些技术类故事称为过程故事、基础设施故事和许多其他名称。但在 Scrum 中，它们都是用户故事，区别只是用户不一样。

7.5 作为冲刺一部分的改进活动

7.5.1 阻塞点和约束理论（TOC）

Scrum Master 的部分职责是帮助解决 Scrum 团队遇到的阻塞点。这些阻塞点可以是任何有碍于团队全身心投入工作的因素。

SM 无需等到有人告知阻塞点或约束存在时才行动。优秀的 SM 会使用下文提及的 POOGI 技术来主动识别约束。

人们通常认为阻塞点是一些阻止工作开展的因素。虽然这是事实，但大多数阻塞点并不会完全停滞工作，它可能会是一个障碍或瓶颈。Scrum 团队，尤其是 SM，可以从约束理论（TOC）中学到很多。约束理论首次由 Eliyahu M. Goldratt 博士在其 1984 年出版的书籍 The Goal 中定义。

Goldratt 博士所提出的前提是，组织和团队可以通过三个指标的变化来衡量：吞吐量、运营费用和库存。吞吐量可以被视为系统产生价值的速率，而运营费用和库存的组合可以被定义为精益中在制品（WiP）。WiP 稍后会做详细介绍。

在达到任何目标之前，还需满足必要的条件，如安全、质量和法律义务。

对于大多数企业来说，目标是通过改进吞吐量、库存和运营费用来实现盈利。

那么，这与 Scrum 中的管理阻塞点和阻碍又有什么关系呢?

7.5.2 聚焦五步法

约束理论的前提是，在目标导向的系统中，目标达成率受到至少一个约束。如果不存在约束，系统的输出将是无限的，而这不切实际。

只有通过增加流经约束的流量，才能提高整体吞吐量，而做到这一点的最好方式是遵循以下五个（简单的）步骤：

1. 识别约束
2. 决定如何利用系统的约束
3. 使其他一切都服从上述决定
4. 缓解系统的约束
5. 如果前面步骤中的一个约束被破坏了，则立刻返回到步骤 1

这五个聚焦步骤旨在确保针对约束的持续改进，这也被称为持续改进过程（POOGI）。

但是，像“利用”、“服从”和“缓解”这些描述意味着什么?

在第二步中，**利用**约束意味着在没有新的投入的情况下，了解如何在约束仍然存在的情况下实现最佳产出。

服从意味着你需要在这些限制范围内来构思如何完成工作。重要的是确保工作在持续进行，即使这意味放慢系统的其他部分。评估工作效率可以通过查看流程而容易地进行观察，也可以通过限制在制品（WiP）来完成。

这样做将为你赢得时间来识别真正的根本原因而不仅是看到症状，并提出一种可以永久消除根本原因的方法。一旦这些措施得以实施，约束将得到**缓解**，同时很可能会立即跳出一个新的约束，因此需要重新开始循环。

重中之重是，谨记要删除在第二步中为"利用"约束而添加的限制，因为这个约束现在已经不复存在。

多数情况下，团体在约束仅被"利用"而没有得到"缓解"的情况下运行了相当长的一段时间，因为修复一些问题需要投入，并且这些问题可能非常复杂和庞大，以至于需要将解决方案作为一个项目或改进计划来实施。

7.5.3 持续改进待办事项列表（CIB）

尽管 Scrum 没有明确确认有一个改进待办事项列表，但最好的敏捷组织都会有一个。

Scrum 指出，改进项应被定义为产品待办事项列表中的条目。但是那些不属于特定的产品待办事项列表的改进项，我们该怎么处理呢？

许多 Scrum 专家认为，一个通用的持续改进待办事项列表在敏捷组织中占有一席之地。如果一项改进明显属于一个产品，那么最好将其记录在产品待办事项列表中。

但是，如果与产品相关的改进项被记录到特定的产品待办事项列表中，那么哪些改进项属于通用的持续改进待办事项列表？

改进产品待办事项列表条目项有三个来源，分别是：

- **非产品特定的审计、CSI 或改善**（Kaizen，经验学习和改进）的方案的结果
- **利用**跨产品或系统的功能**约束**（如，非功能性需求）
- **识别**组织、系统和流程中的**技术债务**，并定义偿还技术债务的方案（在精益和 DevOps 做过，同样也适用于 Scrum 环境）

本书的目的并不是探讨这些方案。但是弄清列表中最后一项（技术债务）也是多有助益的。

7.5.4 技术债务

如果你搜索这个词，你会发现它的定义与软件开发息息相关：

> 技术债[...]是软件开发中的一个概念，它反映了由于选择一个简单或有局限性的解决方案而非耗时更久但效果更佳的方案而导致额外返工所造成的隐含成本。[25]

然而事实却是，所有的组织都有技术债务。简单来说，技术债务是组织中已知的所有错误总和。人们总会说，一旦我们有时间，我们就会解决它。

而现实却是，这往往意味着它永远不会被修复。

---

[25] Technical Debt. (2021). 2021 年 2 月 14 日摘取自 https://en.wikipedia.org/wiki/Technical_debt

这就导致那些半成品、性能低下，受约束的问题，都会慢慢积累，直到它们成为一个巨大的制约因素，会产生重大的负面影响。

切记不要只是纸上谈兵，而是要真正找出时间解决债务。

这就是持续改进待办事项列表的作用所在。说了，就要把它们记录下来。与任何其他产品待办事项列表一样，CIB 必须有一位产品负责人（PO），并且在每轮冲刺周期中都要处理一些工作用于解决 CIB 中的问题。

7.5.5 改进增量

如前所述，Scrum 中有一项规定，在每轮冲刺中至少交付一个增量，CIB 亦是如此。

许多公司选择让所有的 Scrum 团队交付来自 CIB 的增量，也有一些公司拥有专门的 Scrum 团队来专门处理 CIB 产品待办事项列表。无论选择哪种方法，交付改进增量都是构建可持续敏捷业务的方法之一。

8 冲刺期间的其他活动

如前文所述，冲刺从冲刺计划（sprint planning）开始，然后完成一个或多个增量，接下来就是冲刺评审（sprint review）和冲刺回顾（sprint retrospective）。在冲刺过程中，Scrum 团队每天都有状态更新与计划会议，也就是所谓的每日站会。

本书已经对计划会给予了详细介绍，在后续章节将介绍更多有关规划和估算的内容。

下面将介绍其余的冲刺活动，以及一些关于如何在 Scrum 中使用 DevOps 的想法。

8.1 关于每日站会的更多信息

每天的同一时间点，Scrum 团队都会召开固定时间长度为 15 分钟的会议，Developers 以此来规划未来一天的工作。其目的是通过检视自冲刺开始以来已完成的工作来优化团队协作和绩效、规划后续工作、并对剩余工作做出预判。

如前文所述，每日站会允许 Developers 回答三个问题：

- 我昨天完成了什么？
- 我今天准备做什么？
- 我在完成工作的过程中遇到了什么问题？

尽管这三个问题很好回答，也可以提供非常有用的见解，但很多人误以为这就是每日站会的全部内容。

你可以提出的问题不仅仅是这三个；只要是跟团队有关的问题都可以提出。问题完全取决于团队，这些问题也只是开始团队对话的一种手段。

检视和适应是 Scrum 的核心，每日站会也与此相关。

Developers 应该使用每日站会检视完成冲刺目标的进度，并检视冲刺待办事项列表中各项工作完成的趋势。Scrum 团队应该保持警惕，当实际情况和规划出现不一致时可以快速做出反应。

请记住：Scrum 团队是自管理的，这其中大部分就是团队在冲刺过程中对变化、挑战或问题尽快做出合适的响应的结果。

Developers 应该能够立即做出或定义出恰当的响应。

但是，请注意：每日站会不是详细的计划会，它专注于识别问题、依赖关系、挑战和困难。通常情况下，针对这些识别出的内容，团队都可以非常快速的做出明确响应，如果不能，那么“针对问题做出响应”应该成为某个或多个 Scrum 团队成员在每日工作的一部分。Scrum 团队成员通常在每日站会之后立刻进行详细讨论，要么调整、要么重新规划冲刺剩余的工作。

每日站会可以改善沟通，在 Scrum 中它是检视和适应的主要手段之一。

每日站会还可以消除或者尽量减少其他会议的需要，并识别出实现冲刺目标过程中所遇到的障碍。它们还随时向所有人通报情况，确保迅速做出决定，并尽快采取行动。

每日站会归 Developers 所有，由他们来设置议程和形式。尽管 SM 经常负责引导会议的进行，也会对看板和燃尽图进行记录和更新，但是请铭记，这不是 SM 的会议。

这就是说，确保每日站会不会超过 15 分钟是 SM 的职责。如果 Developers 识别出问题而解决方案不清晰时，SM 需要确保会议正常进行。SM 需要询问哪些人需要参与到该问题解决中，以及什么时候可以获得那些解决问题的资源并就这些问题达成一致。当这些问题的答案被找到后，SM 需要确保会议正常进行。这是唯一 Developer 以外的成员可以中断会议的时候。

8.2 评审和回顾

8.2.1 关于冲刺评审的更多信息

冲刺评审在冲刺的最后时间进行，以便检视增量，并在需要的时候更新产品待办事项列表（product backlog，简称 PBL）。

请注意：冲刺评审不是发布或部署阶段关卡，也不是状态和批准会议。评审会旨在征求反馈，促进合作。

如果冲刺包含不止一个增量，在增量完成之后应该立刻被部署或处于可用状态。

冲刺评审再次涉及检视和适应的能力。

在冲刺评审上，Scrum 团队将和干系人一起讨论这次冲刺期间完成了什么，以及这期间是否有未遵照规划的变更。

冲刺评审也将演示所有已完成的工作项，并证明此次冲刺中所有的增量是如何满足 DoD。

然后干系人与 Scrum 团队合作，以确定接下来可能可以优化价值的工作。

SM 确保会议的召开，并让每一个与会者明白其目的。他们还要确保会议时间限制为固定的时间盒内。

本次事件的参与者包含 Scrum 团队和由产品负责人（Product Owner，以下简称 PO）邀请的干系人（比如用户、客户、管理层等）。

评审会的形式由组织根据自身需求来决定，但以下是一些是有经验的敏捷教练关于评审会的建议：

- 由 PO 来阐述产品代表项列表中的事项哪些完成了，哪些未完成
- 由 Developers 来阐述此次冲刺中哪些得到改善，遇到哪些问题，这些问题是如何被解决的
- 由 Developers 来演示已完成的工作，并回答有关增量的问题
- 由 PO 来阐述当前产品待办事项列表。在有需要的情况下，他/她根据目前的进度来规划可能的目标以及预测交付日期
- 全体人员一起协作讨论下一步做什么，这将为后续的计划会提供了宝贵的输入
- 探讨市场和产品潜在用途可能发生的变化，下一步要做的最有价值的事是什么
- 对下一个预期发布产品的特性与功能，审查对应的时间表、预算、潜在能力和市场

冲刺评审的结果产生了被修正过的产品待办事项列表，它定义了下一次冲刺中可能需要开发的产品待办事项列表条目。这个新的产品待办事项列表也可以根据新的时机或者需求进行整体调整。

8.2.2 关于冲刺回顾的更多信息

冲刺回顾会在冲刺评审会之后，下一次计划会之前进行。在会议中，Scrum 团队讨论哪些做得好、哪些需要改进、哪些需要在将来被避免以及下一次冲刺会议的议题。在一个为期四周的冲刺中，冲刺回顾会议通常持续三个小时。如果冲刺的周期短，那么冲刺回顾会议的时间也短。

SM 确保会议的召开，并使团队了解其目的。这为 Scrum 团队再一次提供了检视和适应来识别未来改进的机会。

所有的 Scrum 团队成员都必须参加冲刺回顾会。

SM 鼓励 Scrum 团队改进他们的流程和实践，以便在下一次冲刺中更加高效和愉快的工作。在每次的冲刺回顾会中，Scrum 团队通过计划改进工作流程，或者在恰当的程度且与产品或者组织标准不产生冲突的前提下，调整 DoD 来提高产品的质量。

在回顾会结束时，Scrum 团队应该已经确定了下一次冲刺阶段将实施的改进。实施这些改进也是对 Scrum 团队本身适应性的检视。

尽管改进的实施随时可以进行，但是冲刺回顾会提供了专注检视和适应的机会。

许多 Scrum 团队创建了持续改进待办事项列表，这些列表中包含了无法简单解决的以及每个冲刺都需要改进的条目，因此每一次冲刺中，都应该有产品增量以及改进增量。

9 复杂、大规模的产品待办事项列表

还记得我们之前说过，每一个产品只有唯一的产品待办事项列表，对吧？但是，如果该列表是庞大且复杂的，并且需要多个 Scrum 团队才能实现产品的愿景，那又该如何呢？

无论该怎么做，我们都可以说，将多个团队协调在一个产品待办事项列表里面工作本身就是一件很复杂的工作。

9.1 实现规模化的方法

Scrum 实现规模化的方法有很多，而且许多其他的敏捷方法专门聚焦于敏捷的规模化。

人们已经尝试了许多方法去应对复杂性和规模化，这其中包括：

- SAFe
- LeSS
- Nexus
- Scrum@Scale

我们不会试图在此描述 SAFe 和 LeSS，因为这些方法非常复杂且成本巨大。Nexus 和 Scrum@Scale 是相对简单和轻量级的，并且它们几乎"保持原样"使用所有的 Scrum 实践。

这些方法究竟是什么呢？

Scaled Agile Framework®（SAFe®，规模化敏捷框架）是一套用于在企业级规模之上实施敏捷实践的组织模式和工作流模式。该框架是一套知识体系，包含了针对角色和职责、如何计划和管理工作以及坚持的价值观的结构化指南。

Large-Scale Scrum（LeSS，大规模 Scrum）可被视为产品研发的一个规模化框架，但是它也可被当作组织的规模化框架来使用。它试图通过在产品研发中以一种与众不同的、更加直接的方法解决问题来消除组织的复杂性。需要注意的是，更加直接并不意味着更加简单，特别是从短期来看更是如此。尽管 LeSS 是简单的，但是在采用它的初期可能会比较艰难。

Nexus 是一个用于开发和维持规模化产品和软件研发计划的框架。它以 Scrum 为基础。

类似的，**Scrum@Scale** 同样以 Scrum 作为基础，但是它使用的一些实践可能体现了它的思想是更陈旧的，以及它对于长期的预先规划的依赖是更强的。这自然地让 Nexus 成为一个实现规模化的更简单的选择。

软件研发是复杂的，将这项工作集成到可工作的软件中则需要将很多工件和活动协调好，才能去创造一个完成的结果。此项工作必须是有组织的、有秩序的，同时依赖关系必须被解决并且成果是分阶段完成的。

许多软件研发人员在团队中使用过 Scrum 框架去开发可工作软件的增量。但是，如果多个 Scrum 团队在一个共同的产品待办事项列表和共同的代码库为同一个产品工作时，彼此的工作是会相互影响的，此时他们怎么去沟通呢？如果他们在不同的团队里工作，他们怎么去集成彼此的工作、怎么去测试集成后的增量呢？

当两个团队一起工作时就会出现这些挑战，当三个或者更多的团队一起工作时，事情将会变得更加复杂。

对于使用 Scrum 的许多其他类型项目来说，情况也是如此。所以在大多数复杂的且相互依赖的项目环境中，我们在此讨论的东西也同样适用。

在本书中，我们将会选取 Nexus 作为首要的规模化的方法，因为它可以说是所有方法中最不复杂的。当然，您也可以自由选择适用于您组织的规模化方法。

像外部骨骼[26]一样，Nexus 是关于 Scrum 框架的一套框架。Nexus 将多个 Scrum 团队组合在一起去创建集成增量。它与 Scrum 是一致的，在 Scrum 项目中工作过的人对于它的组成部分都不陌生。Nexus 与 Scrum 的区别在于，在 Nexus 中更多的精力被投入到 Scrum 团队间的依赖和合同合作，以及在每一个冲刺中至少交付一个完成的集成增量。

9.2 从 Scrum 的视角来看规模化敏捷

规模化敏捷最基本的规则是同一个产品不能拥有多个产品待办事项列表。因此，当多个团队为了同一个产品待办事项列表工作时，需要建立一套有效的沟通和协调机制。

一个技巧就是，从一个更高的层面去考虑产品待办事项列表的内容，而不仅仅是具体功能。产品待办事项列表中的条目，之前被称为史诗、主题和功能。可以基于主题将工作分配到不同的团队。此方法可以减少冲突和跨团队依赖的可能性。

在这里需要小心，因为我们将主题描述为产品功能的一个占位符，这有助于我们去组织待办事项列表与优先级排序。请注意，这与 SAFe 中描述的战略主题（Strategic Themes）是不同的，后者是将项目组合与企业的战略联系起来的差异化业务目标。

Scrum 中定义了一些方法来帮助组织实现规模化的、大型且复杂的交付和待办事项列表。在本书的后续章节，我们将更详细地介绍 Nexus。它在一定程度上符合传统的敏捷 Scrum 的扩展观点，但在方法上引入一些微妙变化。

Nexus 引入了一些新的术语。统一的使用这些术语是有价值的，并且可以降低误解的可能性。稍后我们将会介绍规模化与 Nexus 的更多详细信息。

[26] Exoskeleton. (2021). 2021 年 9 月 23 日摘取自 https://en.wikipedia.org/wiki/Exoskeleton

10 可视化管理，Scrum 板和看板板

看板来自于精益的概念。自从敏捷和 Scrum 融入了精益的原则后，当应用 Scrum 时使用看板方法是绝对必要的。由 David J. Anderson 创建的整合了看板和 Scrum 技术的 Scrum 板进一步地证明了看板的必要性。

可视化管理是所有精益和与精益相关的管理技术中的一个关键概念。可视化管理工具比书面信息或口头交流本身传递信息更快、更具吸引力。

如果你不相信，让我们再看一个例子。如果你每天开车上下班，你能准确地说出你停了多少个停车标志或红绿灯吗？但是当你看到红绿灯的时候，你就必须停下来，否则就会造成很多交通事故。这就是可视化管理和信息发射源的威力！

当这类信息以可视化的方式呈现时，团队更容易发现问题并采取行动。使用信息发射源还意味着当问题、缺陷和困难发生时将其暴露出来，从而使团队能够更快、更有效的采取行动。

一种常见的方法是使用 Scrum 板来可视化进度、跟踪团队增量交付的进展以及快速暴露问题和缺陷。

团队通常将 Scrum 板当作看板板，严格来说这是正确的。Scrum 板是最简单的一种看板。因为当看板作为一种方法时，它涉及的内容不仅仅是跟踪进度。因此，Scrum 板和看板是有明显区别的。看板与 Scrum 板相比更复杂一些。

下面是针对两种方法的差异的概述。至于应该采用哪种方法，则应当交由您的团队来决定。

10.1 Scrum 板

在最简单的形式中，Scrum 板是冲刺待办事项列表中的故事和相关任务的可视化展现形式。这是一种很好的技术手段，有助于减少 Scrum 团队在考虑冲刺计划、Nexus 计划或者产品目标时的孤立感或脱节感。

面板通常为四列（或更多列），通常表示如下：

- 用户故事（冲刺待办项）
- 待办
- 进行中
- 已完成

一个软件开发的看板可能包含如下所示的列：

- 用户故事
- 待办
- 进行中
- 测试
- CD/CI（DevOps）流水线
- 完成

请注意，这是一个软件开发看板的例子。如果您在软件开发之外的领域使用 Scrum 来提供价值，那么您的看板肯定是不一样的！冲刺中的 Developers 应该选择适当的步骤来跟踪从“待办（To-do）”到最终“完成（Done）”的进度。

故事以及由其拆分出的任务，通常都写在大小不同的队列卡或便利贴上。

任务通常根据所需的能力分配给某人执行。任务在移动过程中，如果因某些原因，先前分配的人员无法完成该任务（不论什么原因的无法完成）可以重新分配给其他人执行。

请注意，并非所有的故事都是用户需求，有些可能反映依赖性，有些是非功能性需求，有些甚至是优化需求及其相关的任务。

下面的例子展示了一个非常基本的 Scrum 板，其中包含了冲刺中不同故事的泳道。这样就很容易看出一个任务是否属于一个故事、是否已经完成，并使得任务不会在团队成员之间来回流转。

图 16：一个简单的 Scrum 板。该看板展示某个特殊类型冲刺开发周期的任务类型（这里的示例是为机动车制作转向架），并使用泳道将任务链接到相关的 PBI（故事）

图片由 EXIN 基于以下内容创建：Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value*. Amazon Digital Services LLC – Kdp Print Us.

图 17：一个典型的软件开发项目的 Scrum 板，使用彩色的便签来表示该任务分配给谁

图片由 EXIN 基于以下内容创建：Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value.* Amazon Digital Services LLC – Kdp Print Us.

彩色贴纸被用来表示已分配处理该工作的人员。您不一定非要在看板中使用贴纸。

通常只有分配到任务的人被写在工作卡片或便签纸上。卡片上的其他信息可能有：预估完成故事或任务所需的工作量、任务的优先级，或者故事和任务以及任务和其他任务之间的依赖关系。图中没有显示依赖项，稍后在讨论使用 Nexus 进行规模化敏捷时再具体谈论。

10.2 为何使用 Scrum 板

Scrum 板有助于提高可视化和促进良好的沟通。它们使得团队成员能够看到问题在哪里，以便他们能够互相帮助（群策群力），或者因发现之前基于错误假设而更改计划。因此，Scrum 板有助于增进检视、适应性和透明性，并因此促进了团队合作。

它还可以帮助团队致力于冲刺和冲刺中的相关工作，因为每个人都可以通过冲刺看板看到需要所有完成的工作。

创建冲刺看板是冲刺计划会工作的一部分。在这个过程中，团队决定哪些条目（故事）应该包含在冲刺待办事项列表中，并以确保故事的正常交付为目的将其拆分成任务。

Scrum 看板的好处是，您可以立即开始可视化工作。记住，Scrum 并没有规定要使用 Scrum 板，但是使用 Scrum 板对团队来说无疑能带来巨大的好处。

10.3 使用 Scrum 板

最好将创建看板作为冲刺计划的最后一项活动。尽管我们经常说 Scrum Master 负责创建并维护 Scrum 板，但这种说法是不正确的。

Scrum 板属于 Developers，尽管 Scrum Master 可能负责维护它，但这既不是被强制要求的也不是既定的。

Scrum 板属于 Developers 并帮助他们优化他们的工作。

如何创建 Scrum 板呢?

Developers 需要创建一个最能反映他们将要做的工作类型的看板。我们已经举了两个看板的示例，最起码它们应该包括这几列：未开始（或待办事项），进行中（某事）和已完成。

由于产品待办事项（PBIs）通常以故事的形式被加在了冲刺待办列表中，因此它们同时也应该被加在 Scrum 板中。PBIs 将在冲刺计划中被分解为任务，通常具有很多属性，例如持续时长，依赖性，被任命人，完成时间等。将所有关联的任务放在它们所属的 PBI 旁边的泳道中，或者正如我们上面举出来的示例，通过使用相关颜色的便签或卡片。

按上述描述您的初始 Scrum 板应该已经创建完成并且当冲刺计划完成后可以随时使用。

在每日站会期间，讨论围绕着 Scrum 板上所展示的工作事项和每日的变化状态所展开。如果您仍使用三个问题（虽然没在 Scrum 指南中被定义但仍然是可用的），它们将帮助决定哪一张卡片或便签在哪一列中并体现出谁被指派来完成该项任务。看板将映射出那些阻碍任务完成的问题或障碍，通常被称为瓶颈。切记，Developers 致力于完成他们选择的任务而不是在冲刺计划中提前指派的任务。

在软件开发项目中，Scrum 板可以被用来跟踪和报告 bugs。当 bug 被发现，将被加在 Scrum 板中，产品负责人将根据优先级来分析待解决的时间期限。

移动卡片以及指派任务是团队的责任而**不**是 Scrum Master 的。然而团队经常要求 Scrum Master 作为 Scrum 面板的监管人并能够确保其总是反馈当前冲刺的状态或冲刺中的相关迭代。因此，您会经常发现 Scrum Master 参与并执行与 Scrum 板相关的活动。

10.4 看板板与 Scrum 板的区别

看板这个词的字面意思是"可以被看到的卡片"，它起源于丰田（Toyota）生产系统和精益的一个库存计划系统。后来这项技术被用作一个工作调度系统。看板的简易性和可视化可以很好的与 Scrum 和任何敏捷方法结合并且在先前介绍的 Scrum 板中得到印证。

最开始的看板只是一张卡片，它用来指示何时工作站上库存物品不足时可以及时（Just-in-time，准实时）供给，以便下一步工作的正常开展。

类似地，看板任务调度面板给出了下一个任务需要何时完成的提示，这样就可以"及时"[27]的完成工作。

这就是 Scrum 板和看板板的主要区别。看板不仅跟踪进度，而且还跟踪工作是如何在系统中进展的工作**流**。

[27] 原文是 Just-in-Time，准实时。这里考虑并非是讲解看板方法，故用及时来代替。——译者注

10.5 为什么流动很重要?

获得合适的工作节奏和完成工作一样重要。事实上，除非在流程中有合适的节奏，否则无法实现最佳绩效表现。试想一下，如果你所在城市的交通灯一夜之间全都被拆除了，车流会发生什么变化?

道路上车的总量并没有改变，但车流会受到显著影响，最终，一切都会陷入瘫痪。

看板是一种关注整体工作流优化的方法，能够提高整体流程的可预测性、效率和有效性。

记住，Scrum 是建立在经验过程控制理论之上的。经验过程控制的核心是在一个透明的过程中进行频繁的检视和适应。看板实践的主要关注点是通过定期检查所做工作的质量，以及工作在系统或者流程中流转的过程来循环改善。

看板板为工作检查和工作流提供了一定的透明度。团队可以经常调整工作或者工作方法，以便能够高效地获取高质量的结果。

让我们回到 Scrum 板。您是否注意到在图 17 中，Sanjay 有多个任务？假如 Sanjay 可以轻松地在上午完成一项任务，然后在下午完成另一项任务，那么此时不会有太大问题，但如果这两项任务都需要超过一天的时间呢?

这将意味着测试工作将在 Sanjay 这一步挤压，并导致下一个任务将无法完成。Aviaja 必须等 Sanjay 完成测试之后才能部署软件。

在这个场景中，Sanjay 就成了一个瓶颈，不管别人在 Sanjay 之前或之后做了多少工作。团队的可交付成果将由系统中的瓶颈决定，也就是本例中的 Sanjay。因此为了能将 Scrum 板上的工作流最大化，这种与团队其他工作保持同步的完成测试的能力就显得至关重要。

在这种情况下，Sanjay 通常会尝试在两个任务之间进行切换，因为他不想成为瓶颈。任务切换意味着他在同一时间无法专注得做好一件事情，反而是想完成两件事情。这就导致他要么两件事情完成得都不好，要么两个都没有完成。

注意：这是精益和敏捷的核心原则——不惜一切代价阻止任务切换!

现在，如果 Sanjay 是这个团队中唯一的测试人员，那么工作就需要这样规划：在 Dev/Test 列中一次只能有一个任务。

通过这样做，您将确保 Sanjay 一次只专注于一件任务并把它做好，他只有在完成了他当前正在进行的测试任务，并将其交给 Aviaja 部署并提供给用户之后，才能安排另一项任务。

这个概念被称为通过 WIP 限制（work-in-progress limits）来限制在制品数量。通过限制在任何时候所做的测试任务的数量，整个系统会变得更快。

如果 Sanjay 是唯一的测试人员，那么 Dev/Test 列必须有一个数量为 1 的 WIP，该限制适用于 Scrum 板的测试列，以确保整个系统工作良好。或者向团队添加另一个测试人员，然后 WIP 限制可以增加到 2。

这实际上可能意味着，如果达到 WIP 限制，一些 Developers 将停止开发。

阻止某人开展一项新的工作是件好事，这听起来可能有悖常理。实际情况是，他们可以完成正在进行的工作，但一旦完成，他们不能立即开始新的工作，因为这只会使瓶颈更严重！是的，这可能意味着有时候一个团队成员将不会开始某项新工作，但

他们可以帮助其他人完成正在进行中的工作，甚至是将其部署完成，从而让工作可以再次流动起来（在这种情况下，Maria 可能会帮助 Sanjay 进行测试，前提是她没有在测试自己的工作）。如果团队成员不具备解决瓶颈的能力，他们可以通过学习来做到这一点，进而掌握一门新的核心技能。

简而言之：在计划中应该避免瓶颈，这样工作才能流动。精益作为 Scrum 的基础，其主要目标之一就是消除浪费。让我们再次使用车流这个例子。交通信号灯控制车流停止和启动。从本质上来说，交通信号灯使车辆行驶变慢了；但通过这样做，它却使得交通更加顺畅，因此，最终结果是车辆运转更快，更有秩序了。

这是怎么回事呢？

10.6 流动理论

为了确保工作做得更快更好，Scrum 团队需要度量的五个看板（精益）指标是[28]：

1. **WIP**：WIP 是所有已开始但尚未完成的工作。工作未完成的原因不重要，重要的是在 WIP 未被完成之前其它工作无法完成或开始。原因在于他们依赖于在进程过程中的工作项或 Developers 正在忙于 WIP。
2. **周期时间**：从工作项开始到工作项完成所耗费的时间总量。
3. **已用时间**：从工作项开始到当前时间之间的时间总量。该指标仅适用于进行中的项目。
4. **吞吐量**：单位时间内完成的工作项数量。
5. **服务级别预期（Service level expectation，后续简称 SLE）**：用于预测在 Scrum 团队的工作流中，一个给定的工作项从开始到结束需要多长时间。

如果您曾经做过价值流图，那么对这些度量指标应该不陌生。强烈建议以一名 Scrum 实践者的身份来学习精益，这会使您在 Scrum 方面做得更好，并对 Scrum 中为什么有这么多事情是以某种特定方式完成的原因有更深的理解。

为什么这些指标很重要呢？

10.6.1利特尔法则

如果您想理解流的含义，您必须知道利特尔法则，它的公式如下：

$$\text{平均周期时间} = \frac{\textit{平均在制品数量}}{\textit{平均吞吐量}}$$

这是什么意思呢？

简单地说，它证明了在无论在什么时候，您同时做的事情越多，完成整个工作所花的时间就越长。

从逻辑上讲，这是有道理的。当有多个工作项同时进行时，您就会一直在任务之间进行切换，那么完成任何一个任务所需的时间就会更长（已用时间）。除此之外，

[28] Scrum.org., Vacanti, D., & Yeret, Y. (2021). The Kanban Guide for Scrum Teams. Scrum.org. https://www.scrum.org/resources/kanban-guide-scrum-teams

您还不能专注的做事，其结果不仅是完成得工作总量更少（吞吐量），而且完成得也不太好。

看板可以帮助 Scrum 团队通过更好地管理 WIP 来增加循环次数和交付速度。

回到上面示例的 Scrum 板，如果我们推断只有 Sanjay 可以做测试，只有 Aviaja 可以做部署，那么这两列的 WIP 限制都应该是 1。

“开发”列的 WIP 应该为 4：Mubai、Sponono、Yuri 和 Maria。但是，即使四个人可以开发代码，开发的任务数量也决不能超过可以测试和部署的数量。这是瓶颈所能处理的工作数量极限，通过这样做，整个系统将会变得更快。

有人可能会说，开发工作通常比测试工作所需的时间要长得多，可以根据历史数据来调整开发列的理论 WIP 值。实际上，这意味着开发列的 WIP 值可能是 3（比 Developers 总数量少一个），但它永远不会超过可以进行开发的资源数量。

如果 Sanjay 和 Aviaja 可以决定工作的工作节奏，那么这就称为拉动系统。在拉动系统中，当工作在上一个阶段完成后，只有当下一个阶段能够接收时，工作才能被执行。

拉动系统可以防止工作在工作流程中任意的步骤产生工作积压。

团队最理想的状态是精益中的单件流。团队的目标是确保在任何给定的步骤之前，工作都不会中断或等待。这也意味着冲刺中的规划工作流程应该是计划会的一部分。

通过增加 WIP 限制和计划流，我们把 Scrum 板变成了看板板。

看板板还为 Scrum 团队提供了技能和依赖性的清晰视图，可以用最好的方式平衡 Scrum 团队所需的技能和优化工作。在 Scrum 或看板板上使用 WIP 可以很快发现系统中潜在的瓶颈，并允许团队以优化流程的方式调整工作[29]。

Scrum 团队成员经常在相互学习的过程中就掌握了跨职能的技能，这也是团队所需的适应的一部分。Developers 通过在跨职能团队中的虚拟工作变成了 T 型人才[30]。这意味着，在未来 Developers 可以成为测试人员，并通过转换角色来消除系统中的瓶颈。但是请记住，Developers 永远不能测试他们自己的代码/工作。

然而，这是一个**长期**的解决方案。一个更直接的解决方案是增加测试人员的数量，或者看看是否有更多的测试工作可以自动化，并成为 CI/CD 流水线的一部分，这样 Sanjay 和 Aviaja 就可以在一天内完成更多的工作，而无需切换任务。

另外一个短期的解决方案是**群策群力**行为，一个或多个团队成员停止他们手头上的工作并齐心协力的移除流中的阻碍。

---

[29] 在 Eliyahu Goldratt 博士的开创性著作 The Goal 中，还可以阅读更多关于约束理论 Theory of Constraints 的内容

[30] 更多关于 T 型人才的信息可以在下方网站中查询：https://www.exin.com/about-career-path-certifications.

10.7 当使用看板技术时，您的 Scrum 板会有什么变化?

Scrum 板和看板板的主要区别是专注。Scrum 板是对正在进行的工作进行可视化展现，看板板是以最大化流的方式管理工作任务。

在其最基本的形式中，只要将 WIP 限制引入 Scrum 面板，就已经实现了看板的核心原则。

图 18：WIP 限制帮助您如何避免瓶颈并创建一个拉动系统

图片由 EXIN 基于以下内容创建：Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value*. Amazon Digital Services LLC - Kdp Print Us.

让我们假设前面讨论的 WIP 限制是要采取的正确措施，现在您的面板看起来是这样的（参见图 18）。

您将注意到，故事 2 中的任务 2 不应该交给测试（Sanjay），因此需要将其移回 Dev 列。由于它现在占用了 Dev 列中的一个空间，那么 Dev 列的其他任务中应该有一个不能启动，因为 Dev 列的 WIP 限制是 4。

Yuri 不应该启动分配给他的开发代码的任务。那么他该怎么办，什么都不做吗?

正确的答案是，Yuri 应该看看如何帮助 Sanjay 更快地完成任务，同时还要保证任务的质量。他们需要合作并且关注团队整体的承诺。如果他不具备这样做的技能，那么他什么都不做反而对系统中的工作会更好。

在这个例子中，Yuri 没有别的任务可做；但是在大多数冲刺中，Yuri 有可能会处理一个不依赖于瓶颈的任务。

Scrum 建议 Developers 也应该改进他们用来交付增量的系统；从本质上讲，Scrum 建议冲刺规划的工作中应该包含持续改进的部分。在这个例子中，如果 Yuri 不能帮助测试，他也可以从一个不依赖 Sanjay 的测试技能的改进故事（improvement story）的活动开始做点什么，而不是什么都不做。

通过这种形式，Scrum 团队可以确保工作项是完成一个再被拉入一个。所以工作项不会产生不必要的等待。

当出现瓶颈时，团队成员还可以对阻塞或排队中的工作项以及超出团队 **SLE** 的工作项进行快速响应。

Scrum 团队使用 SLE 来发现进行中流的问题，并在未达预期的情况下进行检查和调整。

SLE 包含两部分：消耗的时间，通常为天为单位；以及与预测周期相关的概率（例如，70%的任务应该在冲刺第七天完成）。

SLE 应该基于 Scrum 团队的历史周期时间进行计算，一旦计算出来，有问题的 Scrum 团队应该把它写下来，让每个人都能看到。建议在 Scrum 板上为 SLE 以及产品目标、冲刺目标和 DoD 留一个位置。

如果您是一个新的团队，没有历史数据作为基础，Scrum 团队应该根据团队情况对 SLE 进行预估。

10.8 What are blocked items?

您可能已经注意到上图中故事 2 的任务 3 被**阻塞**了。

尽管在通常的 Scrum 板上指出有哪些问题阻塞了进行中的工作是一个好主意，但是在将 Scrum 和看板结合使用后 Scrum 面板会变得更加重要。

在上面的例子中，一些依赖或问题阻碍了 Maria 完成她的任务。由于任务已启动，因此无法将其移回 To-Do 列；在这里，团队需要了解如何帮助 Maria 处理依赖关系。如果依赖关系是内部的，团队可能会决定调整工作顺序，以便 Maria 的任务可以完成；或者团队成员在这个问题上群策群力，帮助 Maria 分析问题的根因并助其修复。

如果依赖关系是外部的，SM 通常会与外部各方讨论这个问题，以尝试解决它。

尽管之前已经说过，具有阻塞状态的工作项（一个阻塞的标记）不能移回到上一列，但实际情况是，如果什么也做不到，团队可以与 PO 协商，将任务从冲刺待办事项列表移到产品待办事项列表，前提是冲刺目标仍然可以实现。或者实在不行，将其移回到 To-Do 列（不是其他列），希望它能在冲刺的稍晚时间中解决。

如果团队不能解决依赖项或阻碍项的问题，这意味着冲刺目标和 DoD 将无法实现，冲刺就会失败，工作项也应该移回到产品待办事项列表中。

10.9 比较 Scrum 板和看板

下表是使用 Scrum 和看板进行规划对比的*示例*。

表格 2：使用传统 Scrum 板与看板的区别

使用 Scrum 板	使用看板
根据优先级执行工作	基于最大化流动执行工作
通常不能在冲刺期间添加任务，但可以有多个任务同时进行。	完成工作的能力决定了一项任务是否开展。
PUSH 推式体系是可能的	仅 PULL 拉式体系

10.10 使用看板方法

使用看板方法（不是看板）交付产品或服务的价值的方法与 Scrum 不同，这两种方法有一些根本的区别。

除了上述的差异外，使用看板方法还意味着不同：

- 方法
- 角色
- 汇报
- 其他工作方式的

下表列出了这两种方法之间的一些额外差异。

表格 3：Scrum 和看板之间更详细的区别

使用 Scrum	使用看板
限制于一个具有时间盒的事件（冲刺）。	没有时间盒，只要有任务团队就开始工作。
非常具体的职责（PO、SM、Developers）。	没有设置角色或职责，它们根据需要分配。
冲刺中的工作通常是有限的（冲刺待办事项列表）。	并不存在独立的迭代待办列表
在冲刺期间通常不能添加任务。	新的工作事项可以持续的从新客户需求中被拉动，所以任务可能随时变更
团队可以通过做了多少和还有多少要做来衡量绩效（速率和燃尽图）。	因为工作没有被限定固定的时段，所以度量已经完成的工作更符合逻辑（累积流图）。
Scrum 团队在冲刺结束时进行回顾，讨论什么地方做得好，什么地方做得不好，以及将来如何改进[31]。	在迭代后没有一个单独的事件来讨论这个流程变更。但是在精益环境中的团队通常会在每周例会上做一些类似的事情（改善）。

[31] 该方法又称为 Mad Sad Glad Retrospective。——译者注

10.11 在 Scrum 中使用看板会给冲刺带来何种变化?

10.11.1 冲刺

当在 Scrum 中使用看板时，Scrum 的规则是至高无上的。这意味着冲刺的节奏要和以前一样。如果您之前是两周的冲刺，您现在仍然是两周的冲刺，如果您之前是四周的冲刺，您现在也要继续使用这种节奏。

10.11.2 冲刺计划会

基于流的计划会议使用工作流的度量指标作为开发冲刺待办事项列表的辅助工具。Scrum 团队在过去的冲刺中的吞吐量将用于规划团队的后续能力，以便为当前冲刺安排工作。

这听起来很像速率，但现在您将使用 SLE 公式代替过去速率的作用。

10.11.3 每日站会

记住，每日站会是 Developers 的会议。尽管 PO 和 SM 通常也会参加，但除非需要征求他们的意见，否则他们不会参与会议上的讨论。这点在使用看板时也是如此。

在每日站会中，关注重点从完成工作转移到确保工作在系统中流动。瓶颈成为每日站会中需要解决的主要障碍类型。

除了了解正在进行的工作以及当天将要完成的工作之外，Scrum 团队还需要讨论：

- **是否有超过 WIP 限制的?** 如果有，将如何纠正这种情况？具体来说，如何解决被阻塞的工作项?
- **哪些工作进展慢于预期?** 在这里，团队需要查看每个正在进行的工作项的已用时间，哪些工作项已经超过或即将超过 SLE，Scrum 团队可以做些什么来完成这些工作?
- **是否有什么要素或工作项未在面板上显示**，从而影响 Scrum 团队当天完成工作的能力?
- 如果计划中的工作的优先级发生变化，**今天吸取的经验教训是否会对明天的工作产生影响**?
- **是否学到了**任何可能改变 Scrum 团队下一步工作计划的新知识?

10.11.4 评审会

当使用看板方法时，评审会的方式不会改变。

虽然对看板流指标的评审，特别是吞吐量，可能会成为一个新的谈话内容，但最好在回顾会中再详细地讨论这一点。

10.11.5 回顾会

对流指标进行检查和分析有助于确定 Scrum 团队可以对其流程进行哪些改进。Scrum 团队也应该检查并调整"工作流"的定义，以便优化下一个冲刺中的工作流。

团队可以使用累积流图来可视化团队的 WIP、平均周期时间和平均吞吐量。它可以用来改进流程，也可以用来规划团队在下一个冲刺中能完成多少工作。

但是请注意，这里所描述的活动应该属于冲刺过程中检视的一部分，而不仅在回顾会中才发生。

10.11.6 增量

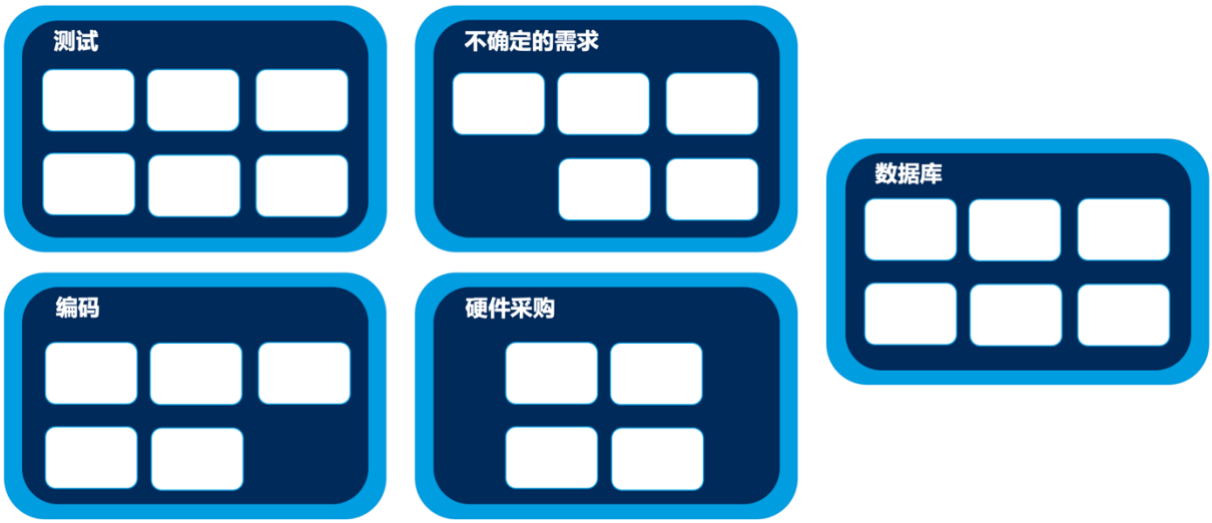
Scrum 要求团队在每个冲刺中至少创建一个潜在的可部署的完成的增量。Scrum 的经验主义鼓励在冲刺期间创建多个增量，并在团队完成工作后部署这些增量，而不仅仅是在冲刺结束时才部署。

因此，获得持续流意味着客户或用户将持续不断地从 Scrum 团队的工作中获益。

10.12 好的 Scrum 板上还应该有什么?

冲刺板或看板板只是 Scrum 团队开会时的信息发射源之一。除了冲刺板或看板板，还可以使用其他工具使得所有工件和相关工作可见，这将促进良好沟通和协调，且促进冲刺执行。其中一些工具已经被讨论过了，比如冲刺目标和冲刺期间要完成的增量，以及相关的工作和记录，比如 DoD 和阻塞工单。但这可能不是团队能够看到的唯一有用的信息。

图 19：使用 KJ 方法对阻塞的标记进行分组



图片由 EXIN 基于以下内容创建：Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value*. Amazon Digital Services LLC - Kdp Print Us.

虽然对阻塞工单及其使用方法已经讨论过，但对于团队来说，当阻塞被消除后也不要把阻塞工单丢掉，在回顾会上，可以用来跟踪已解决的问题或障碍。一些 Scrum 团队保留一个区域来"存放"已解决的阻塞工单，供以后参考（主要是在回顾会期

间），有些团队甚至使用亲和图将它们组织成逻辑上相关的主题，前提是在冲刺期间发生多次阻塞的情况。

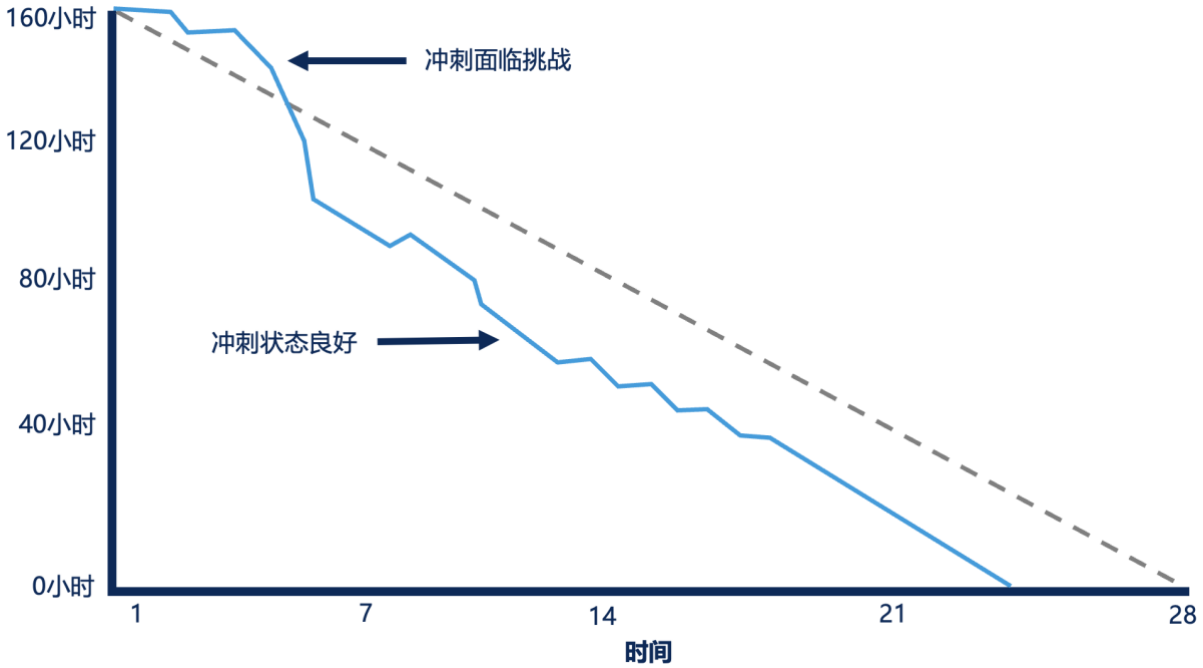
亲和图，也被称为 KJ 方法，它的发明者是日本人类学家 Jiro Kawakita。亲和图是一个图形化工具，旨在帮助组织松散的、非结构化的数据。在上面的例子中，说的就是阻塞工单。阻塞工单被分成有意义的类别，称为亲和集。这些集合围绕一个基本主题将不同的工单联系在一起，用于澄清问题，并为系统性地搜索一个或多个解决方案提供清晰的结构。

10.12.1 燃尽/燃起图

燃尽或燃起图是专门用于跟踪进度的可视化管理工具，但它们也可以测量速率，即在冲刺中完成事情的速度。

冲刺是有计划的，因此需要在一个固定的时间周期内完成，通常是两周到四个周。[32]在敏捷中，您需要确保为冲刺分配的预算和时间盒不变，而可以变化的是您创造了多少价值。由于 DoD 考虑到了这一点，因此冲刺必须至少交付一个符合 DoD 的增量[33]。

图 20：一个典型的燃尽图



图片由 EXIN 基于以下内容创建：Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value*. Amazon Digital Services LLC - Kdp Print Us.

[32] 在实践中，您也许会遇到更长一些的冲刺。但基于 Scrum 指南，最长的冲刺可以是一个月。

[33] 原文是说需要交付一个 DoD，这不符合 Scrum 的交付。所以有理由怀疑作者想说的是"符合 DoD 的增量"。——译者注

当任务完成并对照估计值进行检查时，剩余工作量将被计算并显示在图表上。图形角度可以是向下的，朝着零剩余工作量的方向，这被称为燃尽图；或者角度也可以是向上的，用于显示在两到四周的时间内完成多少任务，这被称为燃起图。

大多数客户更喜欢燃尽图，因为他们更想知道按目前分配的时间周期计算还有多少工作待完成，而不是有多少工作已经完成（这些内容都会在看板板展示）。

在计划期间，图表模板是根据第 0 天到第 28 天的任务数[34]创建的一条直线。上面的例子是一个为期四周、全部工作均已完成的冲刺。这条线表示完成任务的估算速率（虚线=估算速率）。

每天都需要跟踪和绘制实际的进度，如果这条线移动到虚线上方，则表示冲刺没有按计划进行，团队需要进行干预。如果每天的绘图都在虚线以下，则说明冲刺应该可以按时完成并交付所有计划工作。

10.12.2 什么是速率?

速率是衡量一个 Scrum 团队在一次冲刺中能够完成多少工作量的指标，这里没有行业标准。速率是在冲刺结束时计算出来的，方法是将所有已经完成的用户故事的点数相加，然后将这些点数与之前记录的总点数进行比较。在测量速率时，不能将未完成的工作纳入其中，哪怕是 99%完成的工作也不行。

知道团队的速率是计划会的关键输入，这样团队就不会在冲刺期间纳入过多的工作而导致无法完成。

请注意，团队在任意一个冲刺中的速率也取决于他们所做的工作以及工作与团队技能和能力的匹配程度。

对团队速率趋势的评估是回顾会上一个非常有价值的信息输入，可以用于识别需要改进的领域。

一般来说，一个团队的速率应该随着时间的推移而不断提高，具体提高多少有一个参考指标：每个冲刺提高 10%。对于缺乏经验的 Scrum 团队，在冲刺期间，速率低于预期是很常见的。在这种情况下，建议识别可以提升速度的方法；或者在无法提升的情况下，重新评估速率[35]。

注意：PO 可能会在冲刺后重新对产品待办事项列表项进行排序，这个过程会将速率的变化、不断提升的可预测性和弹性[36]纳入思考。

10.12.3 速率和 SLE 有什么区别?

SLE 的概念是早期作为一种预测性指标引入的，类似于燃尽图或燃起图上描绘的理想速率。

其不同之处在于，SLE 的预测不仅基于整个冲刺中应该做多少任务，还考虑了所做任务的复杂程度以及完成任务需要的时间。别忘了，在燃尽图上，您只能在任务完成时将其标记为已完成。如果一个任务花了 5 天时间，完成了 80%，它就不会显示在燃尽图上，但这样就会影响进度和速率的准确性。

---

[34] 此处存疑，应该是剩余工作，但原文是 the number of tasks，此处供读者自行参考。——

[35] 一般是降低速率。——译者注

[36] 原文为 flexibility，可以理解为“拥抱变化”的程度。——译者注

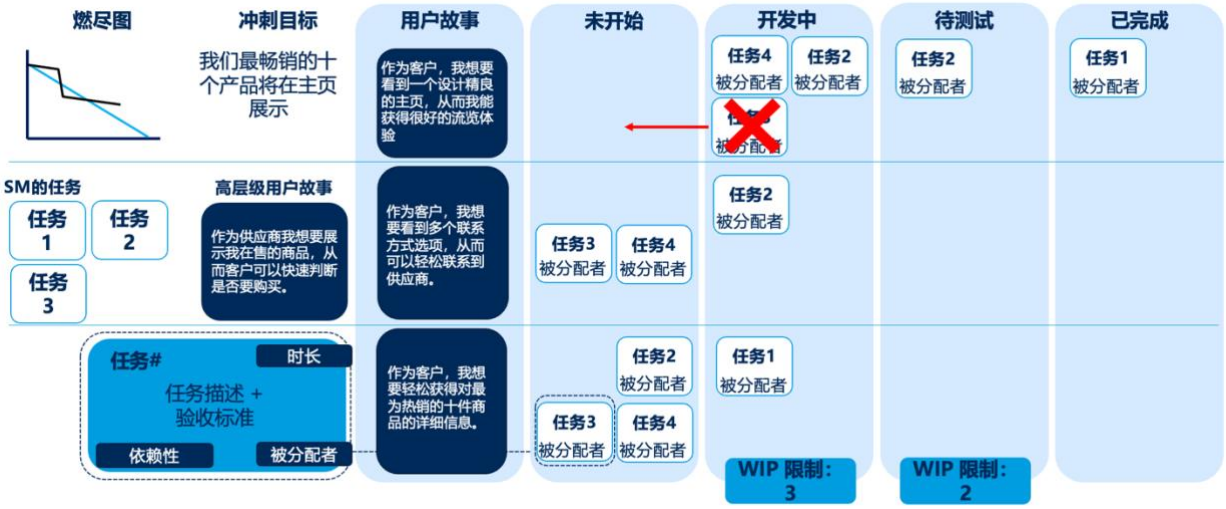
一些团队会因为 SLEs 可以更准确地反映进度而用它在燃尽图上绘制理想的进度曲线。

10.13 信息发射源与会议场所

信息发射源应该是永久性的，团队所有的活动都应该围绕这些发射源来组织，以确保信息在任何时候对团队所有成员都是高度可见的。

一个典型的 Scrum 团队开会时的信息发射源[37]如下图所示：

图 21：一个典型的 Scrum 团队会议空间



图片由 EXIN 基于以下内容创建： Botha, J. (2020). *Scrum Lego-game workbook [Courseware]*. GetITright.

一个典型的 Scrum 团队会议空间除了 Scrum 板之外还包括其他信息发射源，在这里你还可以看到一个燃尽图和一个初始的阻塞项集合，这些阻塞项可以在回顾会上使用 JK 方法进行分析。

[37] 原文是 wall，此处不采纳原文的意思。——译者注

11 规模化 Scrum 的传统视角

由于原始的 Scrum 框架本身未考虑规模化的问题，在使用 Scrum 一段时间后，Scrum 的用户很快就需要解决如何在复杂的大型环境中使用 Scrum 的问题。

11.1 规模化产品待办事项列表

在 Scrum 中有一个规则：一个产品有且只有一个产品待办事项列表。当列表变得庞大和复杂时，您该怎么办?

第一个答案是抵制大型和复杂待办事项列表的诱惑。然而，说起来容易做起来难。

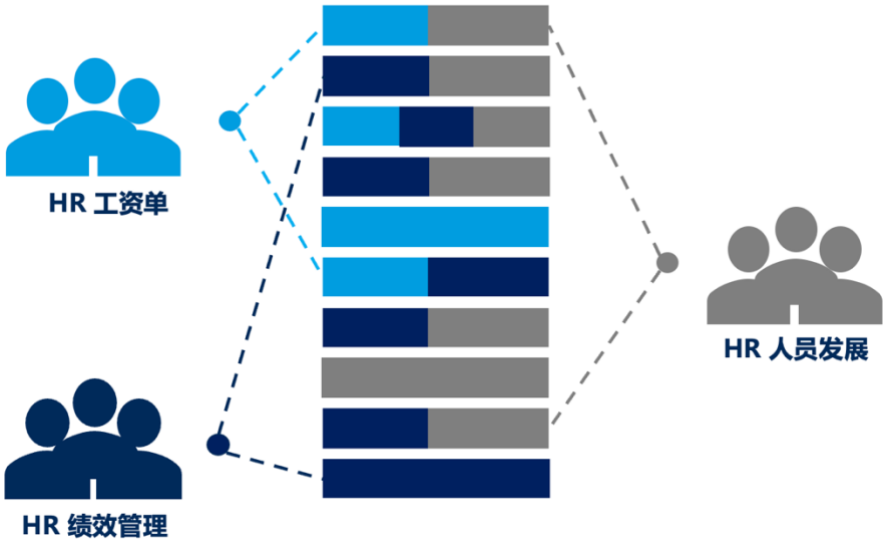
一般的经验法则是，在任何情况下都将产品待办事项列表中的工作项条目保持在 100 项左右。

那么，当有 300 个工作项或 1000 个工作项时，您会怎么做呢？这在非常复杂的环境中，这是很常见的情况。

处理大型和复杂的待办事项列表的第一个技巧是，通过将待办事项列表的事项合并到史诗故事中来降低其复杂度。显然，对于待办事项列表的最靠顶部的事项来说，这样做会适得其反，因为它们需要被尽快处理；但对于待办事项列表最底层的事项，这样做却又对产品管理活动的有效性或效率影响甚微。

寻找低优先级的待办事项，形成统一的主题，并将其重新组合成一个史诗故事。

图 22：创建一个 PO 团队



图片由 EXIN 基于以下内容创建：Botha, J. (2018). *Scrum Masters and Product Owners [Courseware]*. GetITright.

在许多环境中，产品在组织中广泛被使用。在此过程中，即使是同一个产品，但对于不同的干系人群体而言，对产品如何提供客户价值的视角、用法和理解可能会大相径庭。

在这样的环境中，可以创建不同视角的产品待办事项列表，以满足不同干系人的意见和不同优先级的需要。这种对产品的用法和价值的视角差异是传统瀑布项目的最大缺点之一，项目团队专注于针对具体规范提供一项功能，而不会或很少考虑使用该功能的不同干系人群体所期望的结果和价值。

虽然上述内容不是唯一原因，但上述做法也可能导致通过在产品管理组织结构中引入新的角色的方法来扩展产品负责人。

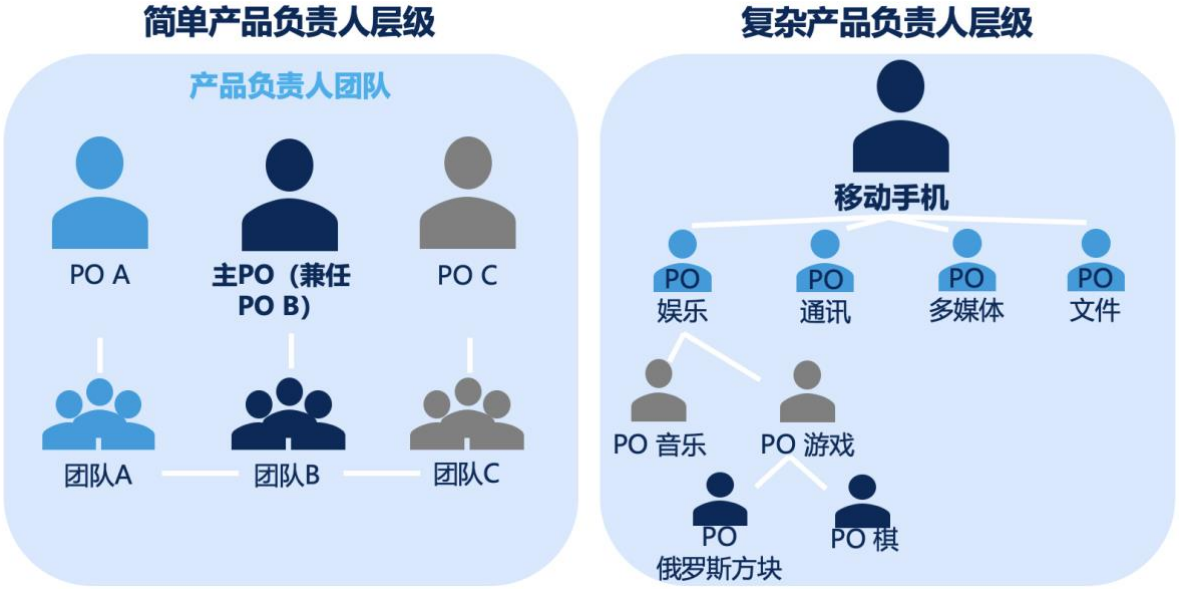
这又意味着什么呢?

请记住，产品负责人的职责是代表业务及其用户。在复杂的环境中，这可能导致大量的工作被压到某个人身上。为了更好地满足干系人的需要，可以审慎地增加多个产品负责人，每个产品负责人都专注于一个/群干系人群体。

然而，这不可避免地会导致一个产品可能有多个产品待办事项列表，这在 Scrum 里是被绝对禁止的。

如何解决这个问题呢?

图 23：构建产品负责人团队



图片由 EXIN 基于以下信息构建：Cohn, M. (2010). *Succeeding with Agile*. Addison-Wesley.

可以如上图中平铺的方式或者分层的方式来组建同时工作在一个待办事项列表的扩展产品负责人团队。在这两种情况下，需要某个产品负责人对产品待办事项列表负责，他被称为"首席产品负责人"。工作如何分配由首席产品负责人决定，但建议该工作是根据干系人群体的需求进行划分，类似于单个产品待办事项列表的多个视角。

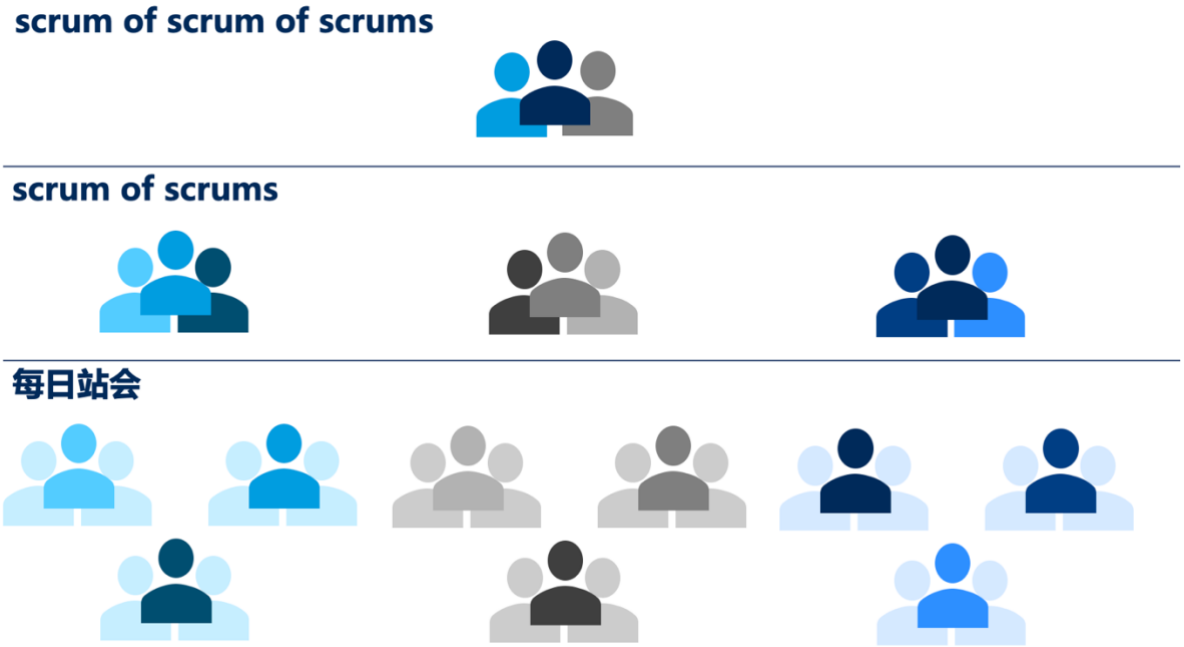
然而 Scrum 明确指出，产品负责人是一个人，而不是一个团队或委员会。因此，上述办法现在存在实施上的困难。

11.2 规模化完成的工作

您需要与多个 Scrum 团队处理同一产品待办事项列表的可能性，将会远远高于需要多个产品负责人处理同一产品待办事项列表的可能性。

协调多个团队工作任务的传统规模化 Scrum 的方式包括将一个或多个冲刺产生的已完成的产品增量转移到一个连贯的发布（版本）中。必要的协调发生在发布计划中或 Scrum of Scrum 中的冲刺阶段。

图 24：通过 scrum-of-scrums 来完成扩展

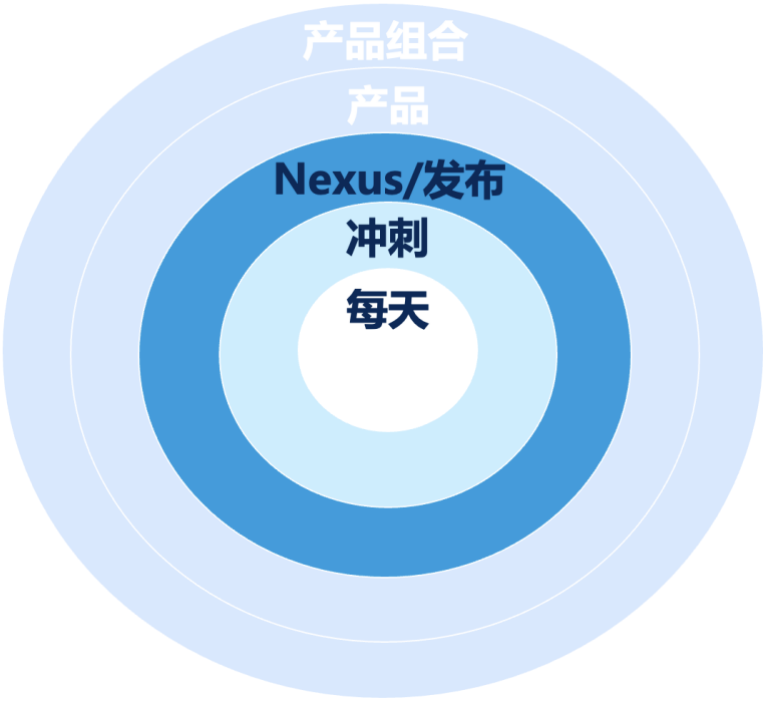


图片由 EXIN 基于以下内容创建：Botha, J. (2018). *Scrum Masters and Product Owners [Courseware]*. GetITright.

敏捷团队主要关注规划级联的三个最内的层级；Nexus，冲刺和每天（见下一个视图）。

发布计划是关于新产品或服务发布所需开发的用户故事或主题。Nexus 或发布计划的目标是确定项目范围、进度和资源。

图 25：规划的三个内侧层级



图片由 EXIN 基于以下内容创建：Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value*. Amazon Digital Services LLC - Kdp Print Us.

我们将在下一章中引入 Nexus 的概念。在这里我们将讲的是一个相对传统的观点，IT 环境中常用的增量交付，即在 Scrum 环境中的"发布管理"。

11.2.1发布管理

一个良好的发布计划应该在整个项目中保持更新，通常在每个迭代开始时进行更新，并且应该始终反映当前发布版本中所期望包含的内容。

下一个层级是冲刺规划。由于冲刺规划比发布计划更为短期，其规划的内容可能更小更具体。

较长规划期的发布管理可能存在显著的缺陷，因为它预示着在项目开始阶段您就知道很多东西。因此您需要提防它最终看起来不应像瀑布式项目规划那样。关于这个问题更详尽的解释可以在附录 B 中找到。

这种方法的另一个问题是，每个组织几乎都发展了其各自的做法。有些非常成功，有些则没有。那些失败的组织，主要是因为他们把 Scrum 变成了 WaterScrumFall。

在渐进式 Scrum 倡导者中，发布计划方法现在已经很大程度上被 Nexus 方法所取代。Nexus 方法现在为扩展 Scrum 提供了一组明确的参考和指导。

Nexus 为使用规模化 Scrum 提供了明确的指导，因此我们在这里不会深入探讨其他的方法。

12 Nexus 与规模化敏捷

12.1 什么是 Nexus?

Nexus 是以可持续方式来进行规模化产品开发的框架，它使用 Scrum 作为其基本构建单元。

它没有改变 Scrum 中的人和概念，也没有对 Scrum 的角色、事件、工件或规则的使用做出任何的调整。Nexus 增强了 Scrum，而不是改变了 Scrum。

从这个意义上说，Nexus 是一个规模化的开发框架。Nexus 冲刺是由多个迭代组成的开发单元，并由此交付一个合并的"增量"，为客户提供价值。

在一个迭代或冲刺中，所有的精力都专注在冲刺的成果上。该成果由 Nexus 的冲刺目标和 DoD 定义。

实际上，尽管我们说，每次冲刺都必须提供潜在的可交付产品。但由于大多数开发环境的复杂性，这通常是不可能的。

工作必须进行排序、执行和交付，同时考虑各种团队的相互依赖和技能，以满足真正的客户成果。因此，冲刺的 DoD 通常无法使用；当客户考虑其需求以及产品或服务需要产出的成果时，增量通常不是可交付的，甚至是不可理喻的。

这在现实中意味着什么呢?

通常，多个 Scrum 团队使用同一个产品待办事项列表来交付价值。在软件开发和许多其他产品中，这意味着使用相同的构建基块（代码库），并直面不同团队工作之间的相同依赖和相互依存关系。不同团队提供的产品增量通常来说必须一起测试和部署，以便解锁对客户的价值。

更复杂的是，Scrum 团队甚至越来越多的不处于同一栋建筑、城市或国家。不协调一致将带来可怕的灾难。

团队需要了解他们所做的工作如何影响他人，其他团队如何依赖于当前团队的可交付成果，以及不完成工作会产生什么后果。此外，需要每个人都对每个团队开发内容的需求、用法与用途有共同理解。

越多的团队参与到这个过程中，情况就越复杂。

由于这些依赖性，因此如果 Scrum 需要扩展，则具有一种可以协调工作、需求的理解以及需求的管理的方法。

此方法应特别满足以下几点：

- **需求。**确保所有团队成员对需求形成全面的理解，以及处理一项需求将如何影响另一项工作。
- **领域知识。**根据团队的技术和业务技能、知识和能力，将已经完成或将要完成的工作映射到不同团队。
- **产品、测试和集成以及质量保证工件。**确保使用统一的方法，将这些组件作为统一的交付物来测试。

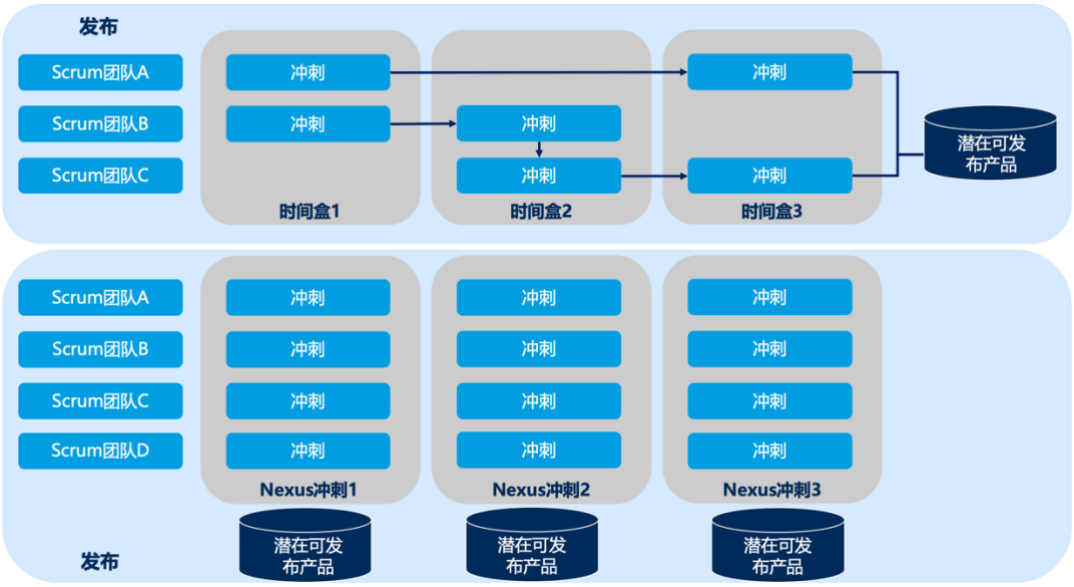
使用 Nexus 等方法将确保工作分配不会像乒乓球那样在团队之间推来推去，并且工作分配过程中不仅考虑了领域知识，还考虑了团队在开发相关或依赖可交付成果方面获得的经验。我们不仅需要一个综合性方法，而且还需要一个优化的方法。

Nexus 通过提供基本框架来实现这一切，团队可以在该基础框架上对其未涉及或者要求的部分进行合理的填充。

12.2 Nexus 与敏捷发布方法的区别

虽然 Nexus 听起来与敏捷发布非常相似，但两者根本的不同是，Nexus 一次只关注一个冲刺或一个时间盒。这是与发布方法的主要区别和出发点。因为 Nexus 认为，项目开始时的对发布的规划是基于不完整的知识，而正是这一点使人们远离瀑布方法（选择 Nexus）。

图 26：Nexus 与发布方法的对比



图片由 EXIN 基于以下内容创建：Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value*. Amazon Digital Services LLC - Kdp Print Us.

许多人会争辩说，发布计划是高层级的，并没有那么详细，它在项目期间会发生变化来适应需求和对项目理解的变化。这当然是发布计划原有的意图。但现实情况是，由于这种方法对于大多数项目参与者来说，与瀑布项目中所习惯的做法非常相似。因此发布计划的变化远远低于支持者所设想或主张的。

许多发布包含了一系列的迭代或冲刺，而 Nexus 始终只关注一个迭代。但可能依然有人会说，这两种方法都有优点。

请记住，发布方法在 Scrum 中并不是必不可少的，它是从其他方法中借用而来的，而 Nexus 是专为 Scrum 设计的。

从上面的图中，您可以看到发布很像瀑布项目中使用的甘特图。

构成若干 Nexus 冲刺的部分的冲刺是单个迭代的一部分，每个冲刺最终都会交付单个增量。而将这些单个增量整合在一起，就成了 Nexus 冲刺的集成增量。集成增量需要满足 Nexus 定义的 DoD 概念[38]。

12.3 使用 Nexus 框架进行规模化

您可以将 Nexus 视为一个框架。它使用现有的 Scrum 方法、角色、工件等。它对 Scrum 进行了增强，从而为业务交付整合的、具有价值的增量。因此，Nexus 团队的关注点更多地放在依赖、互操作性、业务结果及其创造的价值上，而不是仅使用 Scrum 或仅使用 Scrum 作为内部开发的发布方法的情况。

Nexus 团队在整个 Nexus 周期内交付一个完成的增量，并在 Nexus 中的每个迭代也是如此。

要做到这一点，Scrum 角色需要增强。因此 Nexus 创造了一个新的角色：Nexus 集成团队。Nexus 集成团队的职责是协调、指导和监督 Nexus 和 Scrum 的应用，为客户最大限度地提高有价值的成果。请注意，他们不监督 Developers 的工作，因为这违背了自我管理的原则。

这是一个新团队的一个新角色的描述，包括 PO、SM 和每一个 Scrum 团队中的一个代表。该代表为常规 Scrum 团队 Developers 的集成团队成员。

所有 Scrum 团队都应基于相同的产品待办事项列表进行工作，并且梳理活动需要确保所有团队成员都能够很好地看透和了解下一步应处理哪些事项。最好也要确保当团队对待办项排序时，使用了可视化的方式来了解业务需求的即时性。

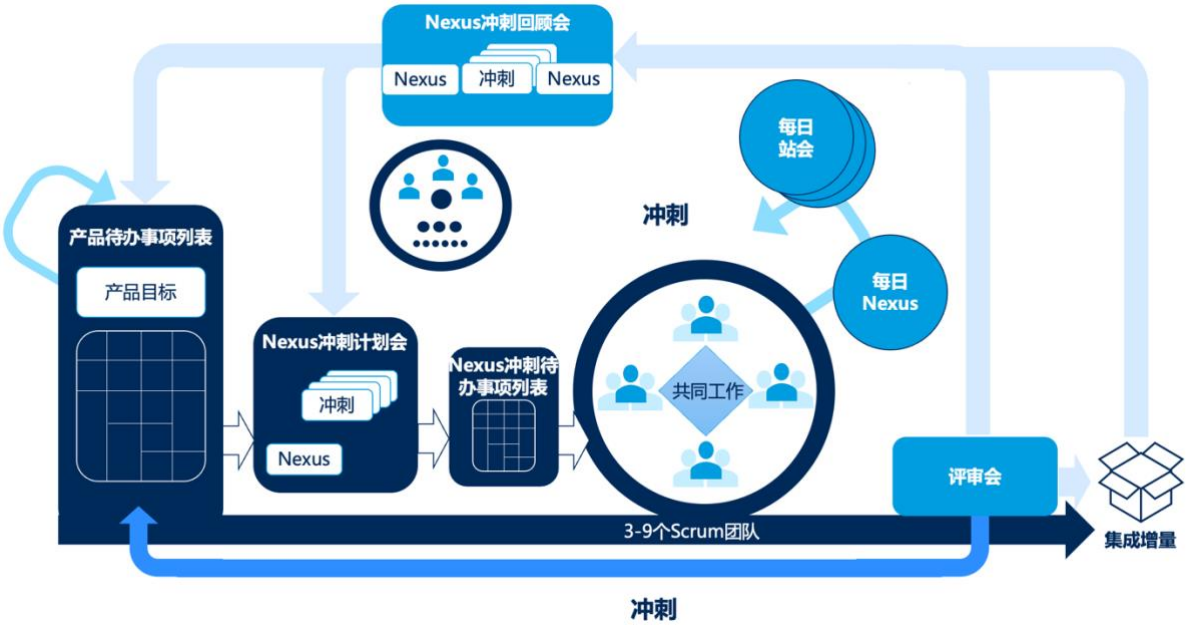
一旦 Scrum 团队选择好了待办事项条目，他们将以惯常的方式执行冲刺，并工作在自己的冲刺待办事项列表上。

Scrum 的任何事件都没有被更换。相反，一些活动现在将成为所有参加 Nexus 的 Scrum 团队的联合事件，或者至少是由所有团队的代表参加。

下图概述了 Nexus 框架的各个组成部分。

[38] 该 DoD 由所有参加 Nexus 的 Scrum 团队共同定义，由 Nexus 集成团队负责确保该 DoD 适用于每个 Nexus 冲刺的集成增量。——译者注

图 27：将 Nexus 与 Scrum 结合使用



图片由 EXIN 基于以下内容创建：Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value*. Amazon Digital Services LLC - Kdp Print Us. 和 Schwaber, K., & Scrum.org (2021). *The Nexus™ Guide – The Definitive Guide to Scaling Scrum with Nexus*. Scrum.org. https://www.scrum.org/resources/nexus-guide.

12.4 Nexus 的流程

注意：本节内容来自 Nexus 指南。

一个 Nexus 冲刺中的所有工作均可由构成 Nexus 的 Scrum 团队的任何成员完成。Scrum 团队的所有成员都是 Nexus 的一部分，他们可以是更大范围的跨职能团队的一部分。团队成员会根据技能和工作项的相互依赖性来挑选待办事项列表条目，以尽量减少不必要的工作分解。

在 Nexus 计划期间，待办事项列表需要梳理，并尽早确定可能会参与处理这些工作的团队。这是一个全新的活动，该活动会添加到产品待办事项列表梳理过程中。

接下来，将进行综合计划会议，它被称为 Nexus 冲刺计划会。每个 Scrum 团队的代表应聚在一起来讨论和审查梳理后的产品待办事项列表、为团队选择工作项，并将这些工作项放入团队的冲刺待办事项列表中。这个综合 Nexus 冲刺待办事项列表依然保留，以综合性方式来审查 Nexus 成果和进行综合的回顾。

保留一个整合 Nexus 冲刺待办事项列表的具体原因是突出依赖关系和相互依赖性！

从以上描述中可以明显看出，团队的 SM 不是代表参加这个活动的理想候选人，因为它与自我管理的原则、以及 SM 作为仆人式领导的角色相冲突。

然后，每个团队都进行自己的冲刺规划，并根据需要与其他团队进行工作整合。团队确保冲刺目标与 Nexus 冲刺计划商定的整体目标保持一致。

在冲刺阶段，开发和测试照常进行，但有些活动可能需要团队之间的协调，以确保在存在依赖的情况下也可以进行集成测试。

因此，最好商定、使用和维护一套共享的测试、集成平台，以及工作方式。

团队仍然执行每日站会。所有团队的代表每天会开会专门讨论集成问题和障碍，并依次向每个团队提供反馈。请注意，Nexus 指南并没有说团队的代表始终是同一个人。从连续性的角度来看，使用同一人这种做法可能有好处。不过，应该选择能够解决 Nexus 当前遇到的问题的最佳人选作为当时的代表。

Nexus 每日站会应在单个团队的每日站会之前完成，这样可以使得来自其他团队的反馈以及对整体工作状态的更好认知成为每个团队日常计划的一部分。

冲刺评审会是以整个 Nexus 为单位进行的。因此，在 Nexus 冲刺评审会上，会展示一个集成好的 Nexus"完成的"增量。产品负责人可以使用此信息做适当的调整并更新产品待办事项列表。

一个 Nexus 冲刺评审会需要每个团队都有代表参与。挑战、问题和经验教训被带回每个团队，然后再由团队进行自己的冲刺回顾会。

然后，每个团队的代表在团队回顾会后再次会面，讨论所有团队所吸取的教训和将要采取的行动。

特别注意：现代项目、变革或改进举措的复杂性很高，所以还应慎重考虑如何实现工作自动化或如何以最佳方式管理复杂的工作流程。

12.5 Nexus 中的新角色

Nexus 集成团队的职责是确保每次迭代都产生集成好的增量。当把多个 Scrum 团队开发的增量集成到一起时，这些增量将实现 Nexus 目标。

Nexus 集成团队是一个 Scrum 团队，由以下组成：

- 产品负责人
  - 记住：单一产品待办事项列表
- Scrum Master
  - 通常一位来自参与团队的成员
- Nexus 集成团队成员
  - 代表参与团队，至少每一个参与团队一名成员代表

如果 Nexus 集成团队成员也承担所属 Scrum 团队冲刺的工作，则必须优先考虑 Nexus 集成团队的工作。作为 Nexus 集成团队的一员，Nexus 集成团队的工作始终优先于 Scrum 团队的工作。这一点很重要，因为他们需要解决与团队如何合作、或处理可能影响 Nexus 成果的问题、或处理影响许多团队的问题，这些工作都会对相关人员产生影响和风险，因此需要优先处理。

虽然 Nexus 团队成员不一定需要在 Nexus 间始终保持不变，但保持稳定的 Nexus 集成团队成员可能有好处的，比如了解工作如何从一个 Nexus 流向下一个 Nexus。但是，团队成员还必须考虑特定 Nexus 的特殊需求，因此 Nexus 团队更有可能随着时间的推移而改变。

Nexus 集成团队负责所有集成问题，他们的工作是确保 Scrum 团队始终将集成团队的角色视为大局的一部分。阻碍 Nexus 的约束和依赖需要重点关注。

该团队（指 The Nexus 集成团队）指导自己的 Scrum 团队在 Nexus 中如何发挥作用。

12.6 产品负责人

基于 Nexus 的团队在单一的产品待办事项列表下工作，因此他们也就要与一个 PO 工作，该 PO 决定了产品待办事项列表。

除了参与以前不存在的 Nexus 活动外，产品负责人的传统职责不会改变。

特别注意：在 Scrum 纯粹主义者中，他们通常不赞成使用之前描述的产品负责人团队架构。然而，将这两种方法结合使用[39]可能也是具有价值的。

12.7 Nexus 集成团队中的 Scrum Master

如前所述，Nexus 集成团队至少需要一名 SM 作为团队成员。虽然这个 SM 也可能是某个 Scrum 团队的 SM，但在复杂的环境中，这可能不是个好主意，尤其是当组织刚开始使用 Nexus 时。

Nexus 集成团队的 SM 需要去辅导团队、引导沟通，并且还要确保 Nexus 成果输出，这与传统冲刺中的职责基本一致。

12.8 Nexus 集成团队成员

开发和价值交付的规模化方法可能与 Scrum 团队所习惯的完全不同，而且很可能需要应用额外的工具和技术来确保工作的正常进行。根据潜在成员在这些方面的能力和技能选择 Nexus 集成团队（Nexus Integration Team，下简称 NIT）的成员，因为除了确保有效使用新的工具和技术外，NIT 还负责实施成功所需的工具、流程和实践。

作为 Nexus 在组织环境中工作方式的专家，NIT 成员也成为使用 Nexus 和相关工具、技术的教练和指导人员。

一个经常被问到的问题是，在 Scrum 中，架构师的角色应该如何改变。Scrum 中的架构师只定义架构原则而不是需要实施的详细操作模型。在 Nexus 中，这些操作模型的定义和实施是 NIT 的责任。

由于 NIT 成员对 Nexus 冲刺中正在发生的事情有更高层次的视角，因此他们也履行了质量管理职责——特别是在迭代过程中对 Nexus 元素的集成。

12.9 Nexus 引入的新事件

新的事件作为组成 Nexus 的冲刺的更高层级的容器[40]，被引入到 Nexus 以确保过程中协调和整合。这些新的事件包括：

12.9.1Nexus 冲刺计划会

在这个 Nexus，会发生什么、谁将为之工作、相互依赖有哪些、优先级是什么？这些都是在 Nexus 冲刺计划会时需要提出的有效问题。

---

[39] 指 Nexus 和产品负责人团队。——译者注

[40] 原文是 higher-level exoskeleton，这里没有取直译结果。——译者注

PO 指导参与者选择每个 Scrum 团队要处理的产品待办事项列表项，但并不决定谁来做这些工作。

但在这一切发生之前，不同 Scrum 团队的代表与产品负责人合作，对产品待办事项列表进行梳理，更具体地说，是基于他们各自的领域知识来调整工作项的顺序。通常，讨论的内容围绕着业务的优先级是什么，以及需要首先解决的依赖关系是什么。产品待办事项列表梳理是业务优先级与实现业务需求所必须满足的技术或实际要求之间的平衡。

如果可能的话，所有 Scrum 团队成员都应参加梳理事件，以最大限度地增进理解和沟通，并最大限度地减少误解和假设。

Nexus 冲刺的目的以及参与 Nexus 将要实现的成果在 Nexus 冲刺计划会的 Nexus 冲刺目标中定义。一旦 Nexus 冲刺目标最终确定并被所有团队理解，每个 Scrum 团队将进行自己的冲刺计划会。

Nexus 冲刺计划会最好能在一个共同的空间内进行，使团队能够持续协作，并在发现新依赖关系、问题和风险时共享这些信息。

由于团队从同一个产品待办事项列表中选择工作项，团队经常会协商并重新评估其团队应该做些什么，以及让另一个 Scrum 团队做这些是否有意义，即使存在依赖关系也是如此。

虽然定义明确的产品待办事项列表将最大限度地减少新依赖关系或需求的出现，但不可避免有些依赖和需求也会在 Nexus 冲刺计划会中发现。

新发现的需求和依赖性应立即将他们可视化。使用通用的可视化规划工具来呈现它们可能非常有帮助。在 Nexus 冲刺计划会中，通过在团队之间移动待办事项列表中的工作项和重新排序工作来减少依赖关系是司空见惯的。

Nexus 冲刺计划会改进了产品待办事项列表的梳理，并且由于 Scrum 团队现在被放置到更大的环境中，使用 Nexus 的好处是随着时间的推移，产品待办事项列表的梳理有了显著改进。

12.9.2Nexus 每日站会

Nexus 每日站会的宗旨是检查当前集成增量的进度，并识别跨团队的依赖或在增量集成过程中遇到的问题[41]。

每个 Scrum 团队都应该适当表达相关内容。“适当表达”这个词隐含的意思是，最了解问题的人应参加 Nexus 每日站会。不要混淆参与 Nexus 每日站会的人与 Nexus 集成团队的成员。Nexus 集成团队成员至少在 Nexus 冲刺期间保持不变；但 Nexus 每日站会的代表可能在 Nexus 冲刺期间都会有所不同。

12.9.3Nexus 冲刺评审会

Nexus 冲刺评审会在冲刺结束时举行，以获取对 Nexus 冲刺中构建的集成增量的反馈。所有单个 Scrum 团队都会与干系人会面，来评审集成增量，同时产品待办事项列表也可能会被调整。

---

[41] 原文是 integration issue，根据上下文和 Nexus Guide 来看应该是集成过程中遇到的问题。——译者注

12.10 Nexus 中的事件

Nexus 事件的持续时间以 Scrum 中相应活动的长度为指导。除了 Scrum 中已有的事件外，新增时间也都是具有时间盒的。

12.10.1 梳理会

规模化的产品待办事项列表的梳理会具有双重用途。它除了帮助 Scrum 团队预测哪个团队将交付哪些产品待办事项列表条目之外，还可以识别团队间的依赖。这种透明度使团队能够监控依赖并将依赖降至最低。

Nexus 对产品待办事项列表条目的梳理会一直持续，直到产品待办事项列表条目能由单个团队独立处理而不会造成过度冲突。

产品待办事项列表中固有的依赖和不确定性将会影响梳理会的数量、频率、会议持续时间和出席者。产品待办事项列表条目经过不同层级的分解，从而将非常大且模糊的需求分解到单个 Scrum 团队可以在一个冲刺内交付可执行的工作。

在整个冲刺过程中，当必要和适当时，梳理就会一直持续。产品待办事项列表梳理将继续在每个 Scrum 团队内进行，以便产品待办事项列表条目准备就绪，能在 Nexus 冲刺计划会中来供团队选择。

12.10.2 Nexus 冲刺计划会

Nexus 冲刺计划会的目的是协调在冲刺中所有 Scrum 团队的活动。产品负责人提供领域知识，指导任务选择和优先级排序。产品待办事项列表应在 Nexus 冲刺计划会之前进行充分梳理，其中的依赖需要被识别、删除或最小化。

有效的 Nexus 冲刺计划会包括三个不同的步骤：

1. 与 Scrum 中一样，**各个团队选择冲刺要做的工作**。这个最好能与其他团队的代表一起协作选择，以避免团队之间的重复工作。
2. **每个团队各自进行常规的冲刺计划会过程**，不同团队之间可以同步进行该工作。
3. **然后随着依赖的发现，团队会重新梳理和组织工作**，以尽量减少不必要的工作中断和团队间的依赖。

在 Nexus 冲刺计划会中，每个 Scrum 团队的代表会验证并调整在梳理会中所创建的工作的顺序。梳理会应要求所有身处 Nexus 中的 Scrum 团队全员参与，以最大限度地减少沟通问题。

产品负责人在 Nexus 冲刺计划会中讨论 Nexus 冲刺目标。Nexus 冲刺目标描述了各个 Scrum 团队在冲刺期间将实现的目标。在了解了 Nexus 的整体工作后，Nexus 冲刺计划会将继续进行，每个 Scrum 团队将进行各自的单独冲刺计划会。Scrum 团队应继续与 Nexus 中的其他 Scrum 团队分享新发现的依赖关系。当所有 Scrum 团队都完成各自的冲刺计划会时，Nexus 冲刺计划会结束。

在 Nexus 冲刺计划会中可能会出现新的依赖。它们应被透明化和最小化。各团队的工作顺序也可以调整。充分梳理后的产品待办事项列表将减少 Nexus 冲刺计划会中出现新的依赖。所有为当前冲刺选中的产品待办事项列表条目及其依赖，应在 Nexus 冲刺待办事项列表中保持透明可见。

12.10.3 Nexus 冲刺目标

Nexus 冲刺目标是为冲刺设定的目标。它是 Nexus 中 Scrum 团队所有工作和冲刺目标的总和。Nexus 应该在 Nexus 冲刺评审会演示其为实现 Nexus 冲刺目标而完成（开发）的功能，以接收干系人的反馈。

12.10.4 Nexus 每日站会

Nexus 每日站会是合适的团队代表（个别 Developers）来检查当前集成增量的状态，并识别集成过程中发现的问题或新发现的跨团队依赖或跨团队影响的事件。

在 Nexus 每日站会中，与会者应关注每个团队对集成增量的影响，并讨论：

- 前一天的工作成功集成了吗？如果没有，为什么？
- 有哪些新的依赖或影响被识别出来？
- 哪些信息需要在 Nexus 中的团队之间共享？

团队使用 Nexus 每日站会来检查向 Nexus 冲刺目标迈进的进度。在 Nexus 每日站会中，应调整 Nexus 冲刺待办事项列表，以反映当前对 Nexus 中 Scrum 团队工作的理解。

然后，各个 Scrum 团队将 Nexus 每日站会中发现的问题和工作带回各自的 Scrum 团队，以便在其单个 Scrum 团队的每日站会中进行规划。

12.10.5 Nexus 冲刺评审会

Nexus 冲刺评审会在冲刺结束时举行，以获得有关干系人对 Nexus 在冲刺阶段构建的集成增量的反馈，并在可根据需要调整产品待办事项列表。

Nexus 冲刺评审会取代了单个 Scrum 团队冲刺评审会，因为整个集成增量是干系人反馈的焦点。在 Nexus 冲刺评审会中，可能无法详细展示所有已完成的工作。可能需要一些技术来最大限度地提高干系人的反馈。Nexus 冲刺评审会的结果是修订后的产品待办事项列表。

12.10.6 Nexus 冲刺回顾会

Nexus 冲刺回顾会是 Nexus 检视和适应自身，并制定下一冲刺需要执行的改进计划以确保持续改进的正式机会。

Nexus 冲刺回顾会发生在 Nexus 冲刺评审会之后，以及下一个 Nexus 冲刺计划会之前。

Nexus 回顾会由三部分组成：

1. 第一部分是一个机会，让 Nexus 中所有团队的代表一起开会，**来识别影响多个团队的问题**。目的是让所有 Scrum 团队都了解跨团队的问题。
2. 第二部分是**每个 Scrum 团队进行自己的冲刺回顾会**，该部分在 Scrum 框架中有描述，此处不在赘述。他们可以使用 Nexus 回顾会议中第一部分提出的问题作为其团队的输入并进行讨论。各个 Scrum 团队应该在各自的 Scrum 团队冲刺回顾会中形成响应的行动来解决这些问题。
3. 最后，第三部分是 Scrum 团队的代表再次开会并就如**何可视化和跟踪已确定的行动达成一致**。这使得 Nexus 能够作为一个整体来调整。

因为以下主题是常见的规模化机能障碍，因此每次回顾会都应该讨论以下主题：

- 是否有工作还未做完？Nexus 是否产生了技术债务？
- 所有工件，特别是代码，是否能够频繁（尽可能每天）成功集成？
- 软件是否能够频繁成功构建、测试和部署，来防止未解决的依赖的过度积累？

针对上面的问题，如有需要，可以问以下问题：

- 为什么会这样？
- 如何消除技术债务？
- 如何防止再次发生？

12.11 Nexus 工件

正如 Scrum 指南中所描述的，工件代表工作或价值，为检查和适应提供透明性和机会。

12.11.1 产品待办事项列表

整个 Nexus 和 Nexus 内的所有 Scrum 团队使用单一的产品待办事项列表。产品负责人对产品待办事项列表负责，包括其内容、可用性和排序。

在规模化敏捷中，产品待办事项列表必须被理解到便于检测依赖性并将之最小化的程度。为了在解决工作过程中起到支持的作用，PBI 中描述的功能通常是细粒度的或*薄片式的*。当 Scrum 团队可以选择的工作项，对其他 Scrum 团队没有依赖、或依赖性最小的情况下就可以完成时，产品待办事项列表被视为已经为 Nexus 冲刺计划会议*准备*好了。

12.11.2 Nexus 冲刺待办事项列表

Nexus 冲刺待办事项列表是各个 Scrum 团队的冲刺待办事项列表的组合。它用于突出显示依赖和冲刺中的工作流。它至少每天更新，更新通常会作为 Nexus 每日站会的一部分。

12.11.3 集成增量

集成增量表示当前 Nexus 完成的所有集成工作的总和。集成增量必须可用且潜在可发布，这意味着它必须符合 DoD。集成增量在 Nexus 冲刺评审会中进行检查。

12.11.4 工件透明度

与 Scrum 一样，Nexus 也是基于透明度的。Nexus 集成团队与 Scrum 团队（可以是 Nexus 或者组织中的 Scrum 团队）合作，确保所有工件都有明显的透明度，并且所有团队成员都广泛了解集成增量的状态。

Nexus 工件的状态将决定 Nexus 期间所做的许多决策，这些决策只有当处于高透明度时才会有效。局部或不完整的信息会导致不正确或有缺陷的决策，而这些决策又会在 Nexus 层面被放大。

增量必须被开发出来，以便于在技术债务成为 Nexus 或正在进行的操作所不能接受的之前，就将依赖性检测出来并将之解决。缺乏完全的透明度将无法有效地引导 Nexus 最大限度地降低风险和最大化价值。

12.11.5 DoD

Nexus 集成团队负责 DoD 可应用于每个冲刺开发的集成增量。Nexus 的所有 Scrum 团队都遵循完 DoD 的定义。只有当 PO 认可增量已经集成、可用且潜在可发布时，该增量方可被视为完成。

单个 Scrum 团队可以选择在自己的团队中使用更严格的 DoD，但不能反向降低团队内部的 DoD 标准。

12.12 Nexus 的核心——Nexus 集成团队

Nexus 由 3 到 9 个 Scrum 团队组成，Scrum 团队共享产品负责人和单个产品待办事项列表。Nexus 集成团队（下简称 NIT）由：

- 一个 PO、
- 一个 SM、
- Nexus 集成团队的成员组成。Nexus 集成团队的成员来自于 Nexus 中的 Scrum 团队。

产品负责人

虽然 PO 是 NIT 的一部分，但他们也担当 Nexus 中的单个 Scrum 团队的 PO 角色。产品负责人确保（负责，accountable）产品待办事项列表的梳理和排序，为干系人实现价值最大化。

Scrum Master

NIT 的 SM 负责确保大家都理解 Nexus 这种方法，并且确保 Nexus 可以正常运行。NIT 通常还引导 Nexus 级别的会议，如 Nexus 每日站会、Nexus 冲刺计划会、跨团队梳理会、Nexus 冲刺评审会和 Nexus 回顾会，或至少他们会支持那些引导会议的人。

Nexus 集成团队的成员

NIT 成员由 Nexus 内的 Scrum 团队的 Developers 组成。NIT 协调 Nexus 内的所有活动。NIT 成员，如 SM 成为各自团队的教练，并共同协调和监督 Nexus 的实施、应用和使用，以确保最佳结果。这点与他在 Scrum 里的职责保持一直。

NIT 成员负责确保至少每个冲刺都会产生集成增量，这些增量是由 Nexus 完成的工作组合而成。

NIT 成员可能会随着时间而改变，具体取决于当前 Nexus 的需要。

NIT 成员通常来自 Nexus 中的 Scrum 团队。一般来说，NIT 成员是兼职的。但一些 NIT 成员很可能是全职成员，尤其是在协调依赖性工作至关重要的复杂的环境中。

但是，NIT 成员应始终将 NIT 成员的角色置于 Scrum 团队成员的角色之上。毕竟整体比局部更重要。

责任

NIT 负责确保至少每个冲刺时都会产生集成增量。这确保了集成产品代表了所有团队的努力成果，而不仅限于单个团队的产出。

规模化软件开发所需要的工具和实践，可能在单个 Scrum 团队不常使用。因此，NIT 应由技术熟练的专业人员组成。但是，集成工作比编码工作涉猎更广。团队如何协同工作和协作是集成工作的重要组成部分。例如，团队如何对破坏性的构建做出反应或实践集体代码所有权，都是成功集成的一部分。NIT 需要拥有兼备"硬"（技术）和"软"（人员）技能的成员。

活动

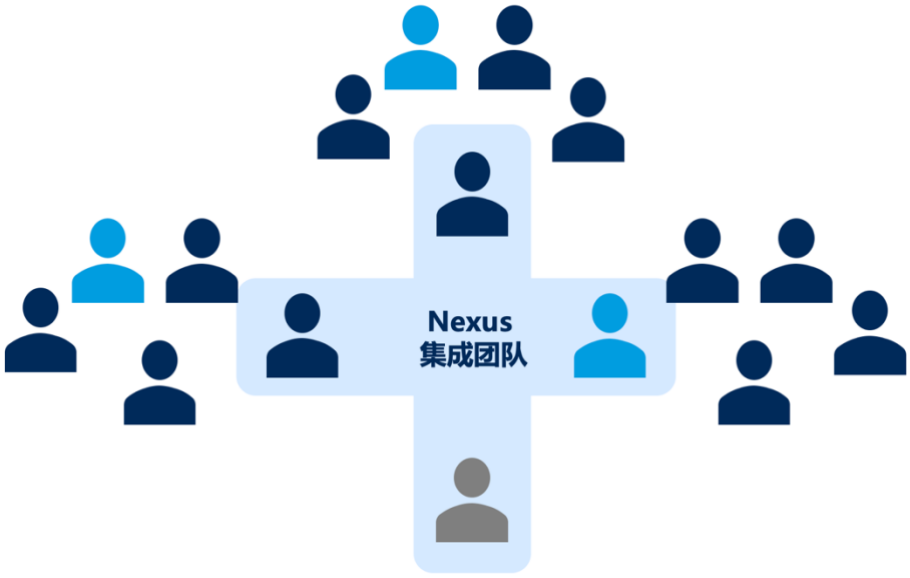
NIT 成员在 Scrum 团队中担任教练。因此，NIT 成员应具备教练所需的技能和特质，使 Nexus 中的 Scrum 团队能够不断改善集成增量的状态。

NIT 每天可能执行的活动包括：

- 帮助协调团队之间的工作
- 尽早提高对依赖的觉察
- 确保大家了解和使用集成工具和实践
- 担任顾问、教练和沟通的桥梁
- 有时酌情协助工作
- 促进共享架构/基础设施
- 不断提供集成的透明度

NIT 中的成员必须是仆人式领导，并使用教练的方式来进行改进。这不是"全明星"团队。如图 28，通过将 Scrum 团队的成员放入 NIT 中，避免了"我们和他们"的鸿沟[42]。

图 28：创建 Nexus 集成团队



图片由 EXIN 提供基于以下信息创建：Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value*. Amazon Digital Services LLC - Kdp Print Us.

[42] 指组成 Nexus 的 Scrum 团队及成员之间产生隔阂。

NIT 由 Scrum 团队的成员组成。NIT 通常不会像 Scrum 团队那样，从产品待办事项列表中拉取工作项。相反，他们是一个教练和提供指导的社区，为 Nexus 中的 Scrum 团队提供服务。由于他们也是这些 Scrum 团队的成员，因此 Nexus 本质上提供了属于自己的领导力。

NIT 作为 Scrum 团队

如果存在严重的集成问题，作为最后手段，NIT 可能会决定他们需要从待办事项列表中拉取工作，并作为 Scrum 团队工作。在这种情况下，他们从现有的 Scrum 团队全职进入 NIT，直到问题得到解决。

他们可能会选择这样做的原因是：

- 这项工作需要的技能只有 NIT 成员具备
- 产品处于未知状态
- Nexus 中的 Scrum 团队暂时无法成功集成

这不是一个可持续的工作模式，因为它意味着 Nexus 未能成功扩展 Scrum，并且正在放缓 NIT 的步伐。最终，如果只有 NIT 能够完成此工作，则规模化已失败。

但是，以这种方式工作的决定应该由包括产品负责人在内的 NIT 自行触发。这将作为 NIT 履行其责任的临时的杀手锏。

让 NIT 作为 Scrum 团队工作被认为是"失败模式"。这应该是一个短期的战术选择。一般来说，NIT 的成员应与 Nexus 中的 Scrum 团队合作，通结对和辅导来转移稀缺的技能和知识。

成员资格

除了产品负责人的角色外，NIT 的成员资格不必是永久性的。事实上，它应该是情境性的。所需的技能类型会随着时间的推移而改变，从而导致团队构成的变化。

Nexus 可能决定邀请来自 Nexus 以外的人加入 NIT。这些人可能包括来自对 Nexus 成功至关重要的领域的代表。如果他们共享相同的集成责任，那么他们参与 NIT 可以带来积极的结果。

关于 NIT 的最后几句话

NIT 不是管理团队，也不应该是具有浅薄专业知识和经验的团队。它是一个专业人员团队，其成员通常来自 Nexus 内部的 Scrum 团队，他们确保实践和工具得到实施、理解和用于检测依赖关系。他们负责确保至少在每一次冲刺时都产生集成增量。他们对集成的关注应包括处理 Nexus 内的技术和非技术问题。

NIT 的作用和 Scrum Master 对 Scrum 团队的作用一致。他们不是作为"主管"的高级经理，而是仆人式领导和教练。他们应该使用团队的自下而上的智慧来解决问题和改进。如果每个团队都能不断为产品开发做出贡献，那么他们就能取得规模化的成功。

12.13 可视化 Nexus 冲刺待办事项列表和跨团队的梳理

Nexus 冲刺计划会的目的是协调单次 Nexus 冲刺中的所有 Scrum 团队的活动。Nexus 冲刺待办事项列表是在 Nexus 冲刺计划会期间创建的，该列表将整个 Nexus 工作都可视化出来，且特别突出了工作间依赖。

Nexus 冲刺计划会需要以结构化的方式完成，来确保工作得到适当协调并考虑所有依赖。

持续的梳理会在复杂的环境中非常必要，因为新的依赖通常只在 Nexus 进行期间被发现。

跨团队 Nexus 梳理面板（Nexus refinement board）被证明是一个有用的工具，来验证和更新 Nexus 中 Scrum 团队已完成的工作。

12.14 跨团队梳理

产品待办事项列表的梳理在 Scrum 中是一个持续进行的活动；请注意，在 Scrum 中梳理不是一个强制的事件。但在 Nexus 中，由于很多团队工作在同一产品（产品待办事项列表）上所产生的复杂性，梳理会变成正式且必要的会议。

梳理共享待办事项列表的重点是充分的拆分产品待办事项列表条目，以便团队能够了解他们可以交付的工作以及在接下来的冲刺阶段交付工作的顺序。该会议使得团队专注于最大限度地减少和消除团队之间的依赖。

特别注意：有些人想知道是什么取代了在 Nexus 中的发布计划。答案是：*没有任何东西取代了发布计划*。但是，共享待办事项列表的梳理可帮助相互理解工作条目的重要性、依赖和顺序，以便在接下来的冲刺中交付。我们来看看下面的问题。

在 Nexus 中，多个 Scrum 团队从产品待办事项列表中拉取工作。因此，有几个关于待办事项列表梳理的新问题需要回答：

- 哪些团队选取了哪些工作?
- 我们如何最好地对跨冲刺和团队的工作进行排序，从而在早期交付价值与风险和复杂性之间取得平衡?

在规模化 Scrum 时，建议这些问题最好通过跨团队梳理会回答，由 Developers 自行找到答案。

每个团队的代表会参加跨团队梳理会。参加梳理会的团队代表应基于正在梳理的工作来选择，而不是基于团队内部的角色。根据所需的技能，让不同的人参加不同的会议可能很常见。

但是，如何确定哪些团队选取哪些工作呢?

对产品待办事项列表项的分解和排序

PO 需要解释待梳理的产品待办事项列表条目。这些通常是比较大的待办项，尚未分解或切分成适合冲刺的大小。这偏离了传统观点，即产品待办事项列表顶部的 PBI 始终会在冲刺计划会开始前梳理完成。延迟这些待办项的梳理，让团队有机会根据梳理过程中出现的技能要求和依赖关系来对产品待办事项列表条目进行分解。

最初的沟通将非常顺畅，且应该聚焦于哪些团队具备开展工作所需的技能。现有的、关于团队专家技能缺失的约束，需要在这个过程中被重点考虑。

一旦 Scrum 团队的代表开始理解 PBI，他们就可以将其分解为较小的待办项，然后将其带回其团队进行常规的产品待办事项列表梳理和冲刺计划。

图 29：创建 Nexus 面板

图片由 EXIN 基于以下信息创建：Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value*. Amazon Digital Services LLC - Kdp Print Us.

请注意，最初的梳理也应考虑依赖。虽然看起来类似于传统的发布计划，但它将确定当前冲刺中可以做什么，以及哪些应放在下一个冲刺。依赖推动了沟通，而不是几个月前向客户承诺的内容。最好也不要尝试在三个冲刺以外的冲刺寻找依赖关系。

因此，理解下一个冲刺的工作流并将其可视化，是每个团队初始梳理活动的一部分。一旦这部分工作完成，NIT 成员需要开始询问："我们如何能够最好地对冲刺和团队的工作进行排序，从而在早期交付价值与风险和复杂性之间取得平衡？"

这是通过在梳理面板（refinement board）上可视化依赖，然后考虑如何最好地管理依赖来实现的。

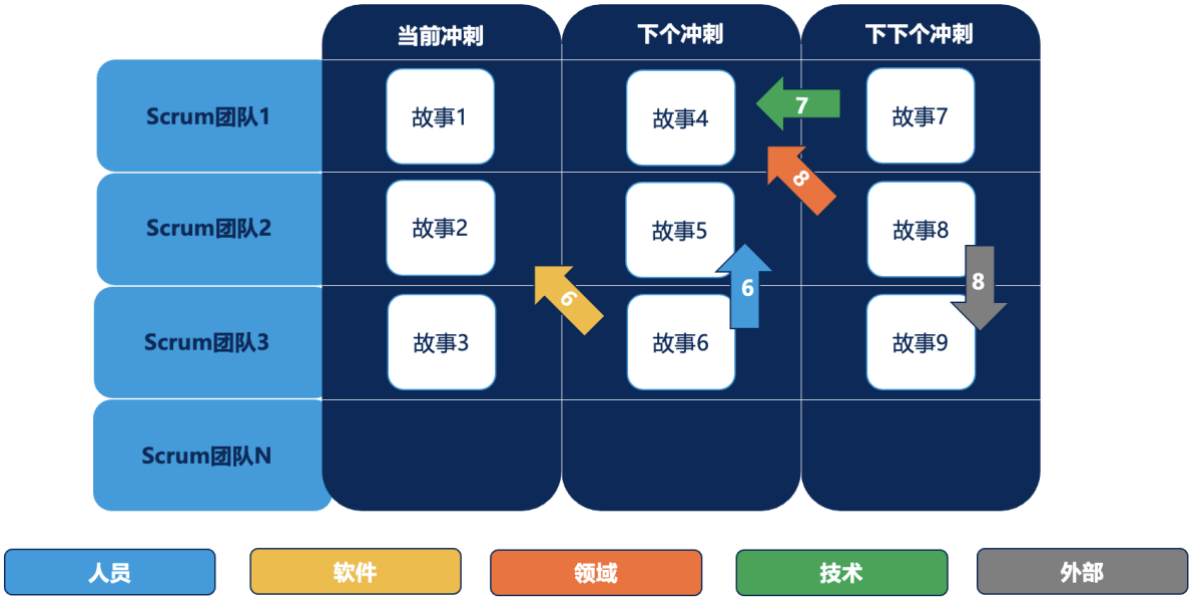
并非所有依赖都具有相似的性质，团队在定义依赖时也应将其分类。

一般来说，可以使用以下类别和子类别用作开始：

- **构建顺序** – 一个待办项在父项完成之前无法完成（可以包括技术、域、软件）
- **人员/技能** – 只有某些人员/团队才能完成待办项
- **外部** – 父项在 Nexus 外交付

依赖由箭头表示，箭头的颜色有助于区分依赖类型。箭头的方向表示父子关系，为了清楚起见，箭头上也可以包含父级故事编号。

图 30：处理分布在 Nexus Scrum 团队间的任务的依赖性



由 EXIN 基于：Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value*. Amazon Digital Services LLC - Kdp Print Us.和 Schwaber, K., & Scrum.org (2021). *The Nexus™ Guide – The Definitive Guide to Scaling Scrum with Nexus*. Scrum.org. https://www.scrum.org/resources/nexus-guide.

至少，请考虑识别外部依赖关系。您可能需要根据不同情景为它们添加颜色。

箭头越多，风险就越高，进行清晰的规划和协调就越重要。依赖突出了未来的工作和梳理工作。梳理工作应该降低依赖或至少降低依赖所带来的影响。

可使用以下通用约定：

🡐水平箭头表示单个 Scrum 团队中的依赖

（通常风险较低）。

🡖向上对角线箭头表示跨时间和团队的依赖

（通常是中等风险）。

🡑垂直箭头表示同一冲刺周期中的跨团队依赖

（这被认为是高风险，因为一个团队的延迟会使整个 Nexus 处于危险之中）。

🡗向下对角箭头表示跨时间的外部依赖

（被视为中等风险）。

🡓向下垂直箭头表示具有外部依赖的冲刺周期依赖

（这被认为是高风险依赖）。

可视化依赖有助于 Nexus 团队了解复杂性，然后 NIT 可以尝试在团队之间重新分配工作待办项，将具有依赖性的工作放在相同的团队中，从而消除依赖，或至少将跨团队的依赖和风险降至最低。

如果 NIT 为了促进在这个冲刺期间交付一个特定的工作而决定在各团队之间分派工作，而这个需要交付的工作太大而无法由一个团队完成时，这就会使得上述的方法无法使用。

以下是将依赖关系和风险降到最低的一些方法：

- 在团队之间移动工作，以最大限度地减少工作交接和跨团队依赖。
- 在团队之间移动人员，以减少跨团队依赖。在一个或两个冲刺中，在团队间重新划分某些技能以最小化团队中的技能依赖性，这可能会显著降低交付风险。
- 通过以不同的方式拆分待办项或重新定义要完成的工作，有可能消除依赖。
- 在 Nexus 面板上工作时，尽量在活动规划中尽早突出所有风险，并尽量使跨团队的冲刺依赖关系尽早暴露出来。

梳理和跟踪依赖不是一次性的事件，需要在整个 Nexus 和冲刺中一直进行。

12.15 更多关于 Nexus 冲刺计划会和每日站会的信息

跨团队梳理后的信息为 Nexus 冲刺计划会提供输入：Nexus 冲刺计划会应考虑当前冲刺中的依赖关系。

依赖关系的信息也应该被用作 Nexus 每日站会期间跨团队同步和管理风险与进度的焦点。

12.15.1 Nexus 冲刺待办事项列表

请记住，可视化冲刺待办事项列表的一种方法是创建 Scrum 面板。为了管理 Nexus 团队之间的冲刺依赖，可以创建 Nexus Scrum 面板，如下示例所示。

在这里，PBI 1 由 Scrum 团队 2 交付，但它依赖于 Scrum 团队 1 交付的 PBI 4，而 Scrum 团队 3 交付的 PBI 2 依赖于 Scrum 团队 2 交付的 PBI 8。

图 31：与团队间的依赖关系共舞

图片由 EXIN 基于以下内容创建：Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value*. Amazon Digital Services LLC - Kdp Print Us.和 Schwaber, K., & Scrum.org (2021). *The Nexus™ Guide – The Definitive Guide to Scaling Scrum with Nexus*. Scrum.org. https://www.scrum.org/resources/nexus-guide.

在 Nexus Scrum 面板中使用 Blocked 列，它有可能指出团队间在当前冲刺中的依赖关系。

在这里可视化是非常重要的，因为通过可视化我们突显出冲刺内的依赖关系以及相关的风险。同时可视化也鼓励我们聚焦在多团队协作与正确的工作顺序上。

在 Nexus 站会中，Nexus 冲刺待办事项列表（Scrum 面板）也是每日站会的焦点并促进对话的产生。

如果 Nexus 没有迭代内的依赖，团队可以选择使用跨团队的梳理面板中的"当前冲刺"栏来表示 Nexus 冲刺 Backlog。在这种情况下，梳理面板就成了每日站会中的焦点。

13 成功实施 Scrum

研究表明，尽管几乎所有组织都采用敏捷方案，但多数认为他们的敏捷成熟度水平低于业内头部水平。但敏捷的优势是显而易见的，比如，更快的软件交付、增强的管理能力以更好地更改优先级，以及提高生产效率和业务与 IT 之间的一致性。

Scrum 是目前组织使用的最为流行的敏捷方法，所以有许多人与你同行。

然而，研究结果也有令人沮丧的一面。

当然，在采用和规模化敏捷的过程中，排名最高的挑战仍然是与组织文化有关。这些挑战包括组织对变革的普遍阻抗，管理层的支持和赞助不足，缺乏领导层的参与，以及存在着与敏捷价值观相悖的组织文化。

然而，挑战的存在也阻挡不了未来将更加敏捷的事实 - 而那些不能接受业务敏捷性的人终将倒落尘埃。

13.1 一切都与组织变革相关

麦肯锡在*《组织敏捷之旅》*（2019）一文中指出，转向敏捷运营模式是很艰难的，尤其是对现有的企业来说。敏捷和许多其他先进实践的敌人是旧的孤岛式运营模式和与之相伴的陈旧的管理方法。

对管理层来说，向敏捷的转变是最艰难的。他们需要放手，相信他们的团队会做正确的事情。将决策权下放至尽可能低的层级员工是敏捷成功的核心，对于一个工业时代的公司来说，要做好这一点是该组织将面临的最具挑战性的努力。

专家们的结论是一致的。企业范围内的敏捷转型是需要的，而且需要通过全面且持续迭代的方式进行转型，以实现敏捷带来的好处。

在文章中，McKinsey 强调了变化的关键要素：

- 人
- 组织结构
- 流程
- 系统与工具

人

在"人"的范畴内取得成功的关键是让领导层接受他们的角色转变。所有管理人员都必须接受培训，将他们的工作重心从监督工作和员工，转移到为团队提供愿景，以便在多学科、自我管理的团队中抓住并利用这些愿景来完成工作。管理工作也是激励和教练形式存在，而不是具体指导如何工作。管理者应该帮助示范工作内容（WHAT），而不是规定如何完成工作（HOW）。

这暗示着，所有团队成员都需要为完成任务做出贡献，并集体决定如何完成任务。如果你在思考关于下一个冲刺会发生什么，那么这正是 Scrum 所建议的。

作为教练工作的一部分，管理者需要更多地关注与帮助员工成长和获取新的技能、能力。实际上，敏捷意味着管理者的工作从管理活动、流程和程序转变为授权成员和组建团队。现在辨识工作的一个好方法是认识到价值流以及人员和流程对创造客户价值的贡献。

知之非难，行之不易。管理者要摆脱已经成为自身发展一部分的泰勒式（Taylorist）管理行为，并接受他们在组织中的新角色——如仆人式领导、使能者（enabler）、教练或导师——是极其困难的。

这也意味着，在组织中，影响力的重要度正在超过岗位名称，这种影响力的自然衍生将在下一节中看到。

组织结构

这种与众不同的业务运作模式，自然衍生出扁平化组织和多技能员工。

现在，一种更加以"任务驱动"或"价值驱动"为导向的方法，将决定团队的模样，并最终决定组织的模样。反过来，这将会影响有关员工队伍规模和定位。

因为团队是作为跨领域的单元、并且通常是临时的结构下进行工作，这意味着组织需要着力简化和分层汇报结构。敏捷组织在默认情况下是去中心化的，且大型企业负责人办公室将成为阻碍而不是助益。

这导致了治理复杂化。因此，治理模型需要简化且高效。我们不能再故步自封了，因为其他组织也发现其中的缘由。我们必须深入地理解风险、建立并不断完善的治理结构、方法与相关控制，特别是基于组织风险的控制。

流程

如今严格的过程与统一的制度常常阻碍企业发展而不是帮助企业发展。流程需要在高层级上定义，同时为团队留下自由的空间，使他们可以自己定义符合高层级指导的工作方式。这意味着每个团队对于同一件事（做事程序）可能有自己的方式（过程）。这意味着只要输出、结果和必要的控制是保持清晰的，就足够了。

组织常用流程和程序作为度量绩效的手段，但是如果流程或程序不一致，则绩效指标应该完全基于结果导向。

系统与工具

管理层应确保组织拥有正确的工具和系统，并将这些工具和系统串联起来，以支持敏捷的工作方式。

这也意味着企业架构将只在原则上集中定义，允许基于随时间变化的需求来设计和演进架构。

如果增量交付不能立即使用，则其毫无用途。自动化、尤其是关乎产品与服务的实现和交付，应该是重中之重。

在软件环境中，这意味着实现了一个自动化的交付管道（比如 DevOps 三步工作法），使测试与集成自动化，从而实现快速而持续交付。

从 IT 基础设施的视角来看，这意味着构建了灵活性和扩展性。而采用云的方法和架构将愈发变得势不可挡。

13.2 促进变革

我们已经在前文描述了 ADAPT 和 ADKAR 模型，它们是促进变革的基础。

采用精益管理原则也是促进渐进式组织变革的一种方式，它们兼收并蓄，不易受到阻碍。

精益管理的一个优势是，它从管理层开始，并由管理层作为仆人式领导的新角色来推动。

这之所以如此重要，因为根据 Prosci 的*《变革管理标杆报告》（2021）（Best Practices in change management benchmarking report）*，管理者抵制变革的主要原因是惧怕失去控制权和权威性。如果管理者拥有更好地做事的新方式，并将其带入到工作中，那么变革在组织内的其他部分就更容易推进。这里有一个附加条件，即管理者需要拥有改变自己行为的意愿，并向员工表明改变他们的行为是安全的。

这份报告继续列举了员工和管理者抵制变革的主要理由，具体如下：

员工	管理层
缺乏认知	惧怕失去控制权和权威性
对未知的恐惧	缺乏时间
缺乏工作安全感	安于现状
缺乏资助	对我有什么好处？
不参与设计	不参与设计

你可以预期最大的阻抗来自那些自认为受到最大损失的人（以为的损失，不是实际的损失），并且你可以预期会有某种形式的派系联盟来反对变革方案。

但是组织内也会有一些人全心全意地接受变革，主要是因为他们对现状不满意。而另一小部分人会支持变革，是因为他们喜欢变革，并为之兴奋。

Chris Musselwhite 和 Robyn Ingram 将这种个人对变革的反应范围称为变革量表，并且他们创造了"变革风格指标"，该指标将个人划分为三个关键类别，代表着个人对变革的看法。

这三个类别以连续图谱的方式存在，分别是：

- 保守派
- 实用主义者
- 发起者

图 32：变革方案中的三种不同类型的用户

图片由 EXIN 基于以下内容创建：Underwood, J. (2013). *Musselwhite and Ingram's Change Style Indicator*. https://innovategov.org/2013/08/15/musslewhite-and-ingrams-change-style-indicator/.

- **保守派**抵制变革，因为他们害怕未知事物及其带来的不确性。
- **实用主义**者则在绝对必要时处于量表和变革的中心。
- **发起者**完全乐于实施变革，并抱有一种"让我们试试看会发生什么"的心态。

尽管这三类人的看法有很大不同，但每一类人都贡献了有价值的见解和特征，这可以使变革方案受益。

保守派的谨慎应在规划中发挥作用。他们喜欢增量的（渐进的）变革，并将以一种不会对业务造成重大破坏的方式转变组织而实现变革。

实用主义者负责促进、合作和调解等工作，而发起者则提供愿景、能量与新颖性。

实用主义者作为团队中心看待其余两方，并经常作为调解人寻求共识。适当的培训或试点项目可有助于实用主义者接受变革。

发起者是变革活动的先知。但不足之处是，发起者作为一位通观全局的人，实施变革的可行性和实用性可能是他们事后才能考虑到的。

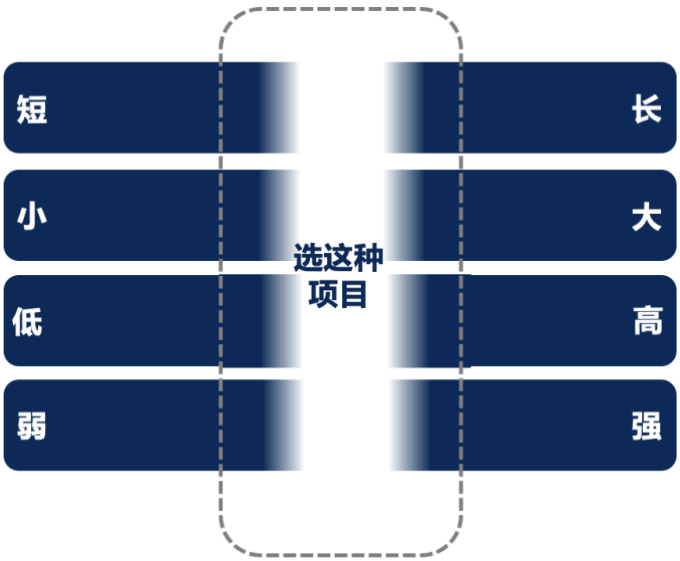
如果组织能够善用发起者实施新想法的意愿，组织就能产生新的业务方式，并提升效率和效益，从而产生更多盈利，为客户带来更多的价值。

13.3 试点实验

一旦知道组织正在采用 Scrum，所有目光都会集中在项目上。一些人期望项目会成功，其他人则希望项目会失败，但是大多数人将持保留看法。

第一批项目的成功至关重要，因此，试点项目应该谨慎选择。建议您在选择第一个试点时，遵循与四个关键指标相关的 Goldilocks principle（金发姑娘原则）。

图 33：选择好的试点项目



图片由 EXIN 基于:Cohn, M. (2009). *Four Attributes of the Ideal Pilot Project.* https://www.mountaingoatsoftware.com/blog/four-attributes-of-the-ideal-pilot-project.

持续时间

选择周期非常短的项目将会导致 Scrum 的反对者声称 Scrum 只适用于短期项目中。

但是，如果您选择了一个周期过长的项目，您将冒着直至项目结束才能宣布成功的风险。

许多传统的管理项目声称自己已经步入正轨，但最终预算超标且工期延期。因此如果 Scrum 项目表现也是如此的话，那么也不具备说服力。

最好选择一个项目，其长度接近您的组织正常情况下的中间值的项目。如果项目超过 6 个月，将其分割为具有明确结果的子项目，并将前三个月作为演示或试点。

这给了团队足够的时间开始熟练使用冲刺进行工作并乐在其中，同时也会看到产品和团队所带来的好处。一个三到四个月的项目通常也足以证明 Scrum 将会在工期更长、规模更大的项目中取得成功。

规模

如果可能，选择一个团队成员已经准备就绪的项目开始。

始终从一个团队开始，即使试点项目最终将发展成包含更多的团队。也可以尝试选择一个最终不会超过 5 个团队的试点项目，即使这样的项目在您的组织中很常见也无所谓。

开始时，不要选择需要在多个 Scrum 团队之间协调工作的项目——这就是在给自己挖坑。毕竟在很多项目中，您可能没有时间将团队从一个发展到五个以上。

重要性

选择一个低重要性、低风险的试点项目可能很具有诱惑力。如果事态变坏，风险会很低；人们甚至可能没有注意到失败。

然而，您的批评者将密切关注您在做什么，并忽视您付诸其中的努力，这对他们来说太容易做到了。

所以相反地，请选择一个风险相对较低的高重要性项目。另外，请记住，为了向 Scrum 转型，团队需择善而从是非常困难的。如果项目不重要，人们可能不会完成分内之事。

与业务发起人保持联系

采用 Scrum 需要在整个业务中进行变革，而不仅仅局限于技术项目资源。拥有一位对敏捷感兴趣的，并且有时间和意愿与试点团队合作的业务发起人是至关重要的。

一位信守承诺并参与其中的业务发起人有助于团队解决分歧，并挑战那些由部门或个人严格执行的墨守成规的流程、实践和管控。产品负责人（PO），特别是在试点项目中，在促进业务以及业务与 Scrum 团队之间的沟通方面发挥着关键作用。在此期间，看到 PO 利用率的增加并不罕见。

类似的，适当地支持显示了管理层的承诺，是促进项目成功的有用工具，尤其是在他们表明情况比预期的要好的时候，这种促进效果更为明显。

13.4 在整个组织中推广 Scrum

试点成功后会发生什么？你如何在企业中扩大 Scrum 的使用？

你需要让组织的其他成员采用 Agile，你可以采取“一步到位”的方法，也可以分阶段实施。尽管许多组织声称“一步到位”对他们有效，但仍建议使用分阶段实施的方法。

有时多个试点也是有用的。这表明敏捷可以做多个不同领域取得成功。你会经常听人提及：“对某某部门来说没问题，但在我们的领域就行不通，因为它们是不一样的”。而多个试点项目的成果就可以很好的反驳这种观点。

你可以使用可扩展的垂直和水平方法：

- **垂直扩展** 在业务单元内扩展
- **水平扩展** 在整个组织范围内扩展

垂直扩展比水平扩展更加容易。因为环境相似，而且学到的许多经验可以很容易地被应用于整个业务，所以教相同业务单元的其他团队如何应用敏捷要容易得多。

试点有助于建立并证明不同团队对 Scrum 的贡献和使用的价值。由于 Scrum 团队是跨职能的，因此业务部门内的不同团队成员应参与试点。

分裂种子法

使用一位或多位来自试点 Scrum 团队的成员作为新 Scrum 团队的成员，意味着当其他成员存在问题、疑惑或单纯想了解或学习一些关于新工作方式的知识时，他们可以立即从这些成员身上获得使用 Scrum 的积极经验。这种方法经常被称为分裂种子法。

增长拆分法

也可以采用一种临时策略：增长拆分。在这种情况下，参与试点的团队成员数增加，老成员会帮助新成员进行两到三次冲刺，然后扩大后的新团队会被拆分，以培养新的团队。这时要培养一个新团队时，既要同时使用经验丰富的成员，也要使用经验不足的成员。

教练

使用教练也是有效扩展 Agile 计划的机制，可以使用内部或外部教练。

教练每周花费数小时与分配给他们的团队在一起，与他们分享如何最佳地使用 Scrum，以及在组织环境中学到的经验教训。因此，内部教练并不是他们所教导的 Scrum 团队的固定成员。

内部教练对情境更为熟悉；然而，外部教练可能更了解 Scrum，并有更多的经验可以借鉴。

建议结合使用本文描述的所有技术在整个组织中推广 Scrum 的使用。

13.5 应对变革中的阻抗

由于种种原因，人们拒绝转型 Scrum。有些人可能使用合理的逻辑和激烈的论据；另一些人则可能通过悄悄破坏变革的努力来抵制。

Mike Cohn 在 *Succeeding with Agile*（2009）中分享了一些相关的深刻见解。本文将探讨其中的一些。

兼听则明，偏信则暗。时常听到人们会基于错误的理解上提出误导的论点或行动。因为他们的阻抗使自己不能获知正确的信息。

比如，有人可能会表示：*你认为没有文档是一个好的主意？好，那我就不给你看任何文档*。当然，在定义敏捷原则时，从来没有认为文档不重要。但是，抗拒变革的人可能会紧紧抓住诸如这样的问题，即使在团队已经针对该问题的处置达成共识的情况下，依然什么都不写下来。

其他人可能会对变革视而不见，墨守成规的工作，然后等待下一次的变革到来，并将 Scrum 扫地出门。

那么，人们是如何抗拒改变自己的行为？您又该如何应对？

人们是如何抗拒的？抵抗情绪是激进的还是消极的？这是我们要弄明白的第一件事情。

- **激进的阻抗**意味着通过采取行动来阻碍或者破坏 Scrum 的实现。
- **消极的阻抗**，另一方面，每当有人失败或忽视行动时，消极的阻抗都是显而易见的。他们觉得如果您忽视其足够长的时间，则其便会消失。

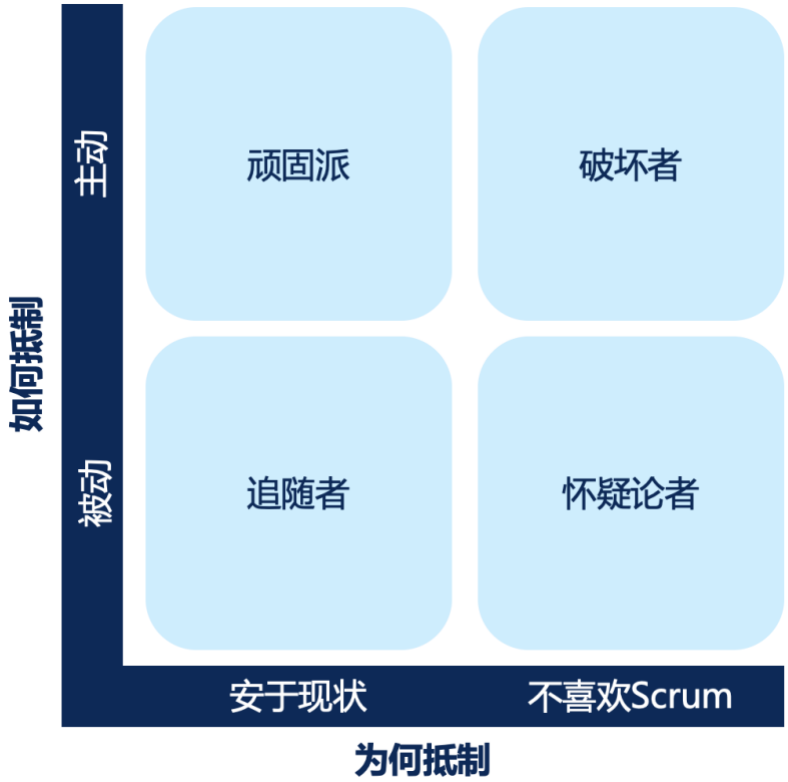
其次，您需要确定人们为什么抗拒变革。通常有两个原因是显而易见的：

- 安于现状:对未知的恐惧
- 不喜欢 Scrum:他们对敏捷和 Scrum 发自内心的厌恶。

他们可能参考一些糟糕的经历（这不一定是他们自己亲身经历的）。亦或是，Agile 所传播的思想对于他们来说太陌生了，因此令人反感。

所以，人们往往有两种抗拒方式，包括两种原因。基于此可以创建一个 2x2 的矩阵，如下所示：

图 34：理解如何处理变革阻抗



图片由 EXIN 基于: Cohn, M. (2010). *Succeeding with Agile*. Addison-Wesley.

每个象限都有一个名称，坐标轴上的标签表示了抗拒者使用的方式。

13.5.1怀疑论者

怀疑论者，他们不同意 Scrum 的原则或实践，同时消极的抗拒转变。怀疑论者会委婉的反对 Scrum 并且频繁的忽视每日站会。他们是试图阻止转变的人，而不是那些发现它有点陌生但依旧愿意尝试的人。

一些用于应对怀疑论者的抗拒时有用的方法，包括：

- 顺其自然
- 提供培训
- 征集同行轶事
- 任命一位怀疑论者
- 推进问题
- 建立意识

13.5.2破坏者

与怀疑论者类似，破坏者抗拒这种转变更多的是出自对于 Scrum 的厌恶。但是，破坏者会采取激进的阻抗手段来达到目的。他们试图破坏转变的努力，可能通过继续编写冗长的前期设计文档等方式来实现其目的。

以下是一些在对付破坏者时行之有效的工具：

- 转型成功
- 重申并加强承诺
- 将他们从现有团队中移除
- 解雇他们（是的，你没看错，有时这就是唯一的解决办法，因为他们会“人身攻击(poison the well[43])）”
- 确保与对的人在交谈

13.5.3顽固派

剩下的则是安于现状的人，他们习惯于当前的做事方式，并且他们的工作往往能为他们在同事中塑造声望与价值。原则上，他们可能并不反对 Scrum 本身，但是他们反对任何能置他们当前处境于风险之中的变革。

顽固派喜欢现状，会激进的抗拒变革。他们甚至会试图团结其他人来阻止这场变革。

下面是如何对付顽固派：

- 调整激励措施
- 营造对现状的不满
- 承认并直面恐惧

[43] 人身攻击谬误，指针对个人的人身攻击或侮辱，而不是直接反驳其提供的理由。可以参考《学会提问》第六章。——译者注

13.5.4追随者

追随者安于现状，消极的抗拒变革。他们不会被变革的前景所激怒，但是他们会尽可能少做与之有关的事务，并希望变革会昙花一现。针对追随者，说服他们的方法是向他们展示 Scrum 已经变为新的现状。

最后，在与追随者打交道时，您可能需要使用的工具：

- 改变团队的构成
- 表扬正确的行为
- 让他们参与进来
- 以身作则
- 识别真正的障碍

13.5.5如何处理各种类型的阻抗

Cohn 的建议简单实用。然而，在决定如何应对那些抵制变革的人之前，你需要考虑组织文化、社会因素和立法环境。

13.6 营造正确的环境

当你开始向 Scrum 转型时，你将面临的第一个挑战是以自管理、跨职能团队的模式进行工作。

考虑到价值观声明，我们可以推测敏捷团队主要以下述方式开展工作：

- 成员必须作为一个 Scrum 团队工作
- 一次只做一件事
- 短迭代工作
- 在每次迭代中交付一些有价值的东西
- 关注业务优先级和结果
- 观察并适应一切

综上所述，我们随即便能发现传统的组织结构、设施和角色所面临的一些挑战。

如果 Scrum 团队是跨职能的，他们仍需紧密配合、定期协作，同时最好为他们创建一个物理的工作空间。

可视化管理是关键，团队成员都应该能看见像 Scrum 板、看板板或 Nexus 板这样的关键信息雷达。因此，最好能拥有开放空间的和可重新配置的工作站。也就是说，有时人们可能需要集中精力，有时需要与其他人会面，而单调的公共区域反而会适得其反。

确保为此目的提供休息设施，但数量不宜过多，从而避免所有人都长期在此工作。日常活动必须在共享的公共区域开展。有时这些设施被称为“洞穴和公共区域（Caves and Common）”。

重中之重，要有足够的空间来进行每日站会。即使在例会完成后，团队的产出物也必须始终保持可见。

如果软件使用 Scrum 板/看板，请确保将其长期固定在每日召开站会位置的墙壁上。

13.7 以虚拟团队和远程团队方式工作

如果说我们从 2020 年的疫情大流行中学到什么，那就是虚拟团队会一直存在。即使在 2020 年之前，许多 Scrum 团队也不在同一地点工作而分布在全球各地。作为虚拟团队一员的人都知道，这会带来一些挑战。

但是，什么是虚拟团队?

虚拟团队是由追求同一目标的人组成的团队，尽管他们不在同一个物理位置。各个团队成员可以在家里或在不同城市或国家的办公室工作。为了使虚拟团队发挥作用，了解虚拟团队的成功公式是很重要的。

目前，对于虚拟团队以及他们应该如何工作的问题有很多不同的看法。但好消息是，作为一个成功的虚拟团队工作的部分内容已经内置于 Scrum 的 DNA 中。帮助多学科团队成为一个充满凝聚力的 Scrum 团队一起工作的原则和实践，同样有助于虚拟团队成为一个有凝聚力的 Scrum 团队一起工作。

眼见为实，让我们观察下虚拟 Scrum 团队以及他们是如何工作的。

- **为所有 Scrum 活动创建一个清晰的流程。**这不需要很详细，但每个人都应该清楚地知道什么时候以及如何发生什么。具体来说包括：
    - 冲刺计划
    - 每日站会
    - 冲刺评审
    - 冲刺回顾
    - 产品待办事项列表的梳理
    - 临时问题解决会议
- **使用一组通用的协作和管理工具。**无论您使用哪种工具，确保所有成员对任务、进度和问题都能持有相同的观点，且这些观点始终有效。
    - 另外，确保协作工具对团队有效，并意识到**面对面沟通**的重要性。重要提示：彼此见面是至关重要的，因为 70%的沟通方式是非语言形式的。

  虽然最好的沟通方式是让人待在一个房间内，但是这不现实。所以，所有虚拟会议都是**基于视频**的团队会议，而不仅仅是语音会议。视频也意味着不能进行多任务处理。
    - 当参与者接入时，**应该在一个标准的、预先确定的地点进行**，这样有助于产生一些有建设性的贡献（场所不应在床上或是火车上）
    - 人们在会议期间需要回答问题或就问题投票，而使用**交互式问题和投票软件**可以增加互动性。
    - 你也可以为团队创造其他空间，**用于日常休闲与活动**，如为团队创建 WhatsApp 群组。
- 确保**所有活动的期望都已经预先设定好**，即使你已经重复无数次了。
    - 团队对这项活动的参与者有什么期望?
    - 参与者能从团队中得到什么?
    - 事件的预期产出和结果是什么?
    - 如果有人的期望不属于标准事件的一部分，那么应该给予机会突出他们的期望。
    - 如果每个人都不遵守会议规则，那么会议将会结束，并且每个人需要重新安排时间。不要视规则如敝屣。

- **一位好的引导者是关键。**这里，Scrum Masters（敏捷专家）可以做出巨大贡献，但是也需要谨记一点——这是谁主导的会议？不在其位，不谋其政。每日站会是属于 Developers 的会议，产品待办事项列表的梳理是属于产品负责人的会议，等等。
    - 避免使用诸如 PowerPoint 之类的工具。它只是一种单向的沟通媒介。在虚拟会议中，所有人的沟通与参与将会变得更加重要。
    - Mike Dwyer 提出了一个很好的建议，他称之为 NOSTUESO 规则**（No One Speaks Twice Until Everybody Speaks Once，除非每个人都讲一次，否则没人可以讲两次）**。
    - NOSTUESO 规则也为人们创造了发言的空间，研究表明，如果一位参与者在前五分钟就发言了，他们再次发言的可能性会增加三倍。
    - 另外，已经证明，当一位女性率先发言后，其他女性的参与程度也会提高。
    - 确保在 Scrum 团队中建立良好的关系和积极而坦率的沟通方式。

14 对其负责-Scrum&Nexus 事件和实践

贯穿全书，我们引用了许多事件或者实践，也暗示了由哪一个角色应该去负责以及应该去执行（例如：SM，Developers，PO）。但是，您在任何地方都找不到一份关于每一个角色需要负责的事情和应该做的事情的完整清单。

本章节包含三个部分，每个部分对应一个角色。在每个部分中，我们绘制和描述了该角色需要执行的和需要负责的 Scrum 事件和实践。

14.1 产品负责人

14.1.1冲刺计划会

	对其负责和职责
参与者	Scrum 团队但由 Developer 拥有
准备冲刺计划会	确保产品待办事项列表已优化并符合当前业务需求。
定义冲刺目标	重申产品目标并确保团队所有成员知道当前的业务需求。在团队中针对冲刺目标达成一致。
定义冲刺待办事项列表	如果需要则提供输入，确保业务需求已反映在待办事项列表中。如果存在意见相左的情况，则将其突出显示，并在可能的情况下与 Developers 协商变更。
评估工作量以及任务拆分	验证假设，提供更高层面的需求和业务的澄清。
创建冲刺待办事项列表	如果被问及则提供输入。
梳理产品待办事项列表	从团队成员处收集输入，更新产品待办事项列表，基于新的见解重新调整列表顺序。

14.1.2每日站会

	对其负责和职责
参与者	Developers 和 SM，有时也需包括 PO 的其他干系人参与。
检视&适应	如果需要则提供输入，以及询问探索性问题。
优化产品待办事项列表	从团队成员处收集输入，更新产品待办事项列表，基于新的见解重新调整列表顺序。考虑交付能力、必要性、时间以及预算。

14.1.3冲刺评审会

	对其负责和职责
参与者	Scrum 团队——由产品负责人认定的关键干系人。有时是由 PO 主持的团队的会议。
启动	确保参与者理解为什么他们需要参与。由于该事件涉及 Scrum 团队以外的干系人，产品负责人带头进行冲刺回顾，
演示和检查	帮助演示交付的增量
获取自己所做事情的反馈	询问探索性问题。
获取环境变更的反馈	记录环境的任何变更。
议题和问题	具体来讲，记录可能会影响交付顺序的依赖。
优化产品待办事项列表	从团队成员处收集输入，更新产品待办事项列表，基于新的见解重新调整列表顺序。考虑交付能力、必要性、时间以及预算。

14.1.4创建和维护产品待办事项列表

	对其负责和职责
收集需求	接受收集到的需求。某些时候，PO 自行收集需求；但是，这件事情也可以由客户和用户、分析师或者团队其他成员完成。不过，PO 的确需要仔细检查需求，并且常常会在将需求添加到产品待办事项列表之前将此需求拿回去做反复优化。
待办事项列表的首次梳理	考虑初始需求，从团队成员处收集输入，基于业务需求以及各其他的见解填充产品待办事项列表。考虑交付能力、必要性、时间以及预算。
待办事项列表的持续梳理	从团队成员处收集输入，更新产品待办事项列表，基于新的见解重新调整列表顺序。考虑交付能力、必要性、时间以及预算。

14.1.5冲刺回顾会

	对其负责和职责
参与者	Scrum 团队。该事件由 Developers 主导，从其他角色处获得输入。
检视	参与但不主导
适应	参与但不主导
学习分享	尽管学习主要是聚焦于团队内的，PO 可以选择向共同为同一待办事项列表工作的其他 Scrum 团队分享学习。

14.1.6Nexus 冲刺计划会

	对其负责和职责
参与者	参与 Nexus 的团队成员，以及 PO。
准备冲刺计划会	Nexus 永远不会使用跨产品待办事项列表，PO 扮演着更加主导但仍然是指导的角色，帮助团队参与 Nexus 的计划中来。在现实中，PO 是 Nexus 的召集人，因此，PO 在组建 Nexus 和让团队参与 Nexus 方面扮演着重要的角色。他们的角色与正常的冲刺计划会类似。
启动	PO 也是 Nexus 负责人以及召集人；因此，计划和执行启动会也是 PO 的责任。

14.1.7Nexus 冲刺评审会

	对其负责和职责
参与者	参与 Nexus 的团队成员，以及 PO。
执行	尽管 Nexus 冲刺评审会的内容和形式与冲刺评审会类似，但是它需要协调和计划。在 Nexus 中，没有单独的冲刺评审会，只有 Nexus 冲刺评审会。该会议由 PO 所拥有和运行。

14.1.8Nexus 冲刺回顾会

与冲刺回顾会一致。

	对其负责和职责
参与者	Scrum 团队。该事件由 Developers 主导，从其他角色处获得输入。
检视	参与但不主导
适应	参与但不主导
学习分享	尽管学习主要是聚焦于团队内的，PO 可以选择向共同为同一待办事项列表工作的其他 Scrum 团队分享学习。

14.1.9Nexus 产品待办事项列表梳理

梳理 Nexus 产品待办事项列表不仅是一个持续的活动，也被定义为一个 Nexus 事件。该事件由 PO 执行和协调。

14.2 Scrum Master

特别注意：SM 赋能 Scrum 团队成员，但是他们从不主导任何事件或者行动。他们只是引导、教练和协助团队。

14.2.1冲刺计划会

	对其负责和职责
参与者	Scrum 团队，但是由 Developers 主导。
准备冲刺计划会	帮助协调和组织此会议。
定义冲刺目标	作为引导者、调停者来帮助解决冲突。
定义冲刺待办事项列表	作为引导者、调停者来帮助解决冲突。
评估工作量以及任务拆分	作为引导者、调停者来帮助解决冲突。
创建冲刺待办事项列表	尽管冲刺待办事项列表属于 Developers，但是他们时常请求 SM 帮助他们去创建、文档化和维护该列表。
梳理产品待办事项列表	作为引导者、调停者来帮助解决冲突。

14.2.2每日站会

	对其负责和职责
参与者	Developers、SM，有时也需 PO 或者其他干系人参与。
检视&适应	推动对话、在需要时提供指导和调停。
处理阻碍	SM 的一项主要职责是帮助 Developers 去处理阻碍；通常来说，这涉及到推动会议、获取协助、资源或者技能，以及与受影响的干系人经常沟通并采取后续行动。
跟踪进度	及时更新信息发射源通常是由 SM 完成。这不是他们的责任，但是此项任务通常由 Developers 委派给 SM 代为处理。
对团队进行教练工作	SM 的一项主要职责是指导和教导团队，以确保他们精通 Scrum 的实践并且熟悉 Scrum 最新的发展状况。
梳理产品待办事项列表	引导对话、在需要时提供指导和调停。

14.2.3冲刺评审会和冲刺回顾会

	对其负责和职责
引导	推动对话、在需要时提供指导和调停。Scrum Master 充分地准备冲刺回顾。

14.2.4创建和维护产品待办事项列表

	对其负责和职责
收集需求	在此项活动中，SM 向 PO 提供支持。
待办事项列表的首次梳理	推动对话、在需要时提供指导和调停。
待办事项列表的持续梳理	推动对话、在需要时提供指导和调停。

14.2.5Nexus

并非 Scrum 团队中的所有 SM 都会参加 Nexus 相关的事件，例如：Nexus 计划会、Nexus 评审会和 Nexus 回顾会。但是至少需要一个 SM 以团队 SM 的身份参与到 Nexus 集成团队中去。这个角色会投入很多时间协调和优化团队间的活动。尽管这可能不是一个全职工作，但是在复杂的 Nexus 中最好将其设置为永久并聚焦的角色。相对于 Scrum 的团队工作，需要始终将 Nexus 的工作置于更高的优先级。

14.3 Developers

14.3.1冲刺计划会

	对其负责和职责
参与者	Scrum 团队，但是由 Developers 主导。
准备冲刺计划会	冲刺计划会是 Developers 的工作，他们通常将准备会议的任务委派给 SM 完成。
定义冲刺目标	冲刺目标由 Developers 与 PO 负责协商确定。
定义冲刺待办事项列表	由 Developers 从 PO 处获得输入，有时在 SM 的帮助下选择的冲刺待办事项。
评估工作量以及任务拆分	Developers 从 PO 处获得协助及输入，并完成工作量的评估以及任务的拆分。此过程中也需要 SM 的帮助。
创建冲刺待办事项列表	Developers 创建冲刺待办事项列表，简化了此项任务，他们通常请求 SM 去监控该列表。因此此时只有一个人负责列表的更新。
梳理产品待办事项列表	Developers 应该协助 PO 去优化产品待办事项列表。但是，这是一个持续的活动；它通常是冲刺计划会的最后一项任务，Developers 会基于计划会期间学习到的东西做一下快速的优化。

14.3.2每日站会

	对其负责和职责
参与者	Developers、SM——有时也需包括 PO 在内的干系人参与。
检视	Developers 汇报进度、发现依赖和阻碍。
适应	Developers 决策如何处理阻碍和依赖。如果需要外部协助，他们可以请求 SM 和 PO 去推进，以期达到一个好的结果。
跟踪进度	跟踪进度是 Developers 的责任；该任务通常委派给 SM 处理。
梳理产品待办事项列表	任何需要优化产品待办事项列表的新信息都需要在 Developers 暴露问题时完成。PO 负责产品待办事项列表的优化工作。

14.3.3冲刺评审会

	对其负责和职责
参与者	Scrum 团队以及由 PO 识别出的干系人。有时候由 PO 来主持。尽管是团队会议。
启动	由于此事件涉及 Scrum 团队之外的干系人，因此冲刺评审会由 PO 主导。
演示和检视	Developers 向其他干系人展示他们的成果并回答问题。
获取对已完成以及环境的反馈	值得一提的是，冲刺评审会并不是一个签到会，而是一个从干系人，特别是客户和用户处获得反馈的机会。干系人时常会分享新的需求或者环境变更，而这些是 Scrum 团队之前无法意识到的！反馈也可以时不时地暴露出议题和问题；然而，此实践却是不提倡的，因为局面可能很快就失去控制，
梳理产品待办事项列表	当有了新的信息、见解和反馈，且这些信息在团队的脑海中还是新鲜的时候，就是优化产品待办事项列表的绝佳时机。

14.3.4创建和维护产品待办事项列表

	对其负责和职责
参与者	PO，Developers 从旁协助。
收集需求以及产品待办事项列表的优化	Developers 通常不参与初始的需求收集，但是通常在冲刺中与用户交互的过程中、或者在计划和执行冲刺的过程中，会发现新需求。新的需求需要提交给 PO，通常是作为产品待办事项列表优化的一部分。

14.3.5冲刺回顾会

	对其负责和职责
参与者	Scrum 团队。
检视	许多技巧可以用；其中一个即是询问如下三个问题：什么地方做得好，什么地方出问题了，以及哪些地方可以改进。在这里，冲刺的过程中收集的关于议题和阻塞的分析可能会有所帮助。
适应	类似的，团队可以询问他们自己应该改变、改进和避免哪些来创造改进的方法——并将这些转换成可操作的计划。
学习分享	分享从其他 Developers 学习到的经验会有所帮助，在规模化实施中更是如此。

14.3.6Nexus 冲刺计划会和协调

	对其负责和职责
参与者	参与 Nexus 的团队成员，以及 PO。
协调	Nexus 冲刺计划会将由所有参与的 Scrum 团队的代表完成，该团队被称为 Nexus 集成团队。这通常包含从参与的 Scrum 团队中借调的有经验的 Developers。

14.3.7Nexus 冲刺评审会

在 Nexus 中，要进行联合冲刺评审。当组成集成增量的工作都被评审后，联合冲刺评审就完成了。最起码，Nexus 中所有 Scrum 团队的代表需要展示他们工作成功并获取反馈。Nexus 冲刺评审会的其他工作方式与冲刺审会是一致的。

	对其负责和职责
参与者	Scrum 团队的代表、PO 以及选定的干系人，包括客户和关键用户，前来参加 Nexus Scrum 评审会。

14.3.8Nexus 冲刺回顾会

所有 Nexus 相关的 Scrum 团队必须首先完成一个常规的 Scrum 回顾会，然后团队的所有成员作为一个整体在 Nexus 中重复这个活动。如果将分享的经验教训推广到所有团队，可以获得巨大收益。

	对其负责和职责
参与者	Nexus 中所有的 Scrum 团队的所有成员。

14.3.9梳理 Nexus 产品待办事项列表

优化共享的待办事项列表聚焦在在产品待办事项列表条目（PBIs）进行充分的分解，让团队理解他们可以交付的工作，以及交付工作的顺序（在接下来的冲刺阶段）。该事件也让团队聚焦于最小化以及消除团队间的依赖。

	对其负责和职责
参与者	至少应该由 Nexus 集成团队的成员参与该活动；但是，我们建议更多的 Nexus 中的 Scrum 团队的代表也来参与。

附录 A - 其他敏捷方法

这是一篇内容汇编，出自一些被注明出处的公共资源。本文的唯一目的是供读者准备 EXIN Agile Scrum Product Owner 和 EXIN Agile Scrum Master 认证。

自 2001 年的《敏捷宣言》发布以来，许多组织已经采用了不同的敏捷方法来帮助他们团队实现他们的目标和可交付成果。这其中最著名的方法是 Scrum，被广泛用于各类企业。早期，这些敏捷实践仅被 IT 部门采纳和应用。相较之下，当前敏捷方法广泛应用于各种不同的非 IT 业务领域。

除 Scrum 之外，本附录还将概述和简要解释最常用的敏捷方法：

- 水晶（Crystal）
- 极限编程（XP）
- 动态系统开发方法（DSDM），现在称为 ABC
- 大规模 Scrum（LeSS）
- 规模化敏捷框架（SAFe）
- 看板

有一点请务必牢记，尽管这些方法都不同，但它们有三个共同点：开放式沟通，团结与灵活性。所有这些方法都共享这些要素，因为这些共同点是敏捷宣言的核心。

水晶方法[44]

水晶家族是一组轻量级的敏捷方法。之所以它们被认为是轻量级的，是因为这些方法认为过程是次要的，重点应放在其它元素上。

这些元素是：

- 人
- 互动
- 团队
- 技能
- 才能
- 交流

20 世纪 90 年代末，Alistair Cockburn 提出水晶方法论。它们是作为 Cockburn 进行的研究结果被创建的，该研究表明，Developers 并没有使用他们想要使用的正式的方法，但他们仍然在交付成功的项目。Cockburn 对一些重要因素做了区分：

- **方法论：**一组元素（如实践、工具）
- **技术：**技能领域（如开发用例）
- **策略：**组织必须如何行为的定义

依照他的研究结果，Cockburn 对团队成员的行为做了如下定义：

- 人是沟通的生物，最好使用面对面、亲自参与的、实时的提问与回答。
- 随着时间推移，人们很难始终如一地采取行动。
- 人充满变数，日复一日，因地而异。
- 人们通常都想成为良好公民，善于环顾四周，主动作为，做'任何需要'做的事情保障项目可以正常运作。

水晶方法论分为 8 种不同的颜色。颜色用于表示方法论的"权重"。该列表从适用于最多 6 人的项目的透明水晶方法开始，一直到适用于对人类生命有潜在风险的、关键任务项目的水晶钻石或水晶蓝宝石。

同时还有一点需要注意，水晶家族的所有成员都有 7 个共同的体系特征。这些特征包括：

- 频繁交付
- 反思改进
- 近距离或渗透式沟通
- 人身安全
- 专注
- 与专家用户建立便捷的联系

频繁交付

定期发布正在开发中的软件/产品的迭代。

[44] Crystal Methods 免费阅读. (2021). 2021 年 9 月 23 日摘自 https://en.wikiversity.org/wiki/Crystal_Methods

反思改进

项目团队应邀不时地思考如何改进他们的流程。这不仅提供了开发任务中的短暂休息，还有助于提升团队的效率。

近距离或渗透式沟通

沟通必须贯穿整个项目团队。为了确保发生，项目团队应该同处于一个空间内，提倡有助于项目进展的沟通。

人身安全

团队中的每个人必须感觉处于一个开放和安全的环境，可以自由发言。消极的回应，如当有人提出问题或建议时受到嘲笑，必须被阻止。这是因为人们必须能够相互信任彼此，任何成为消极情绪目标的人都可能不再主动参与。

专注

保持专注是成功的关键。当在水晶方法论中提及专注时，它指代两件事情：进度与方向。进度意味着为项目中的任务投入足够的时间，以确保取得进展。而方向代表项目前进的方向。

与专家用户建立便捷的联系

项目团队必须能够联系到正在开发的新功能的真实用户。通过这种方式，这些用户可以回答任何与新功能有关的问题并帮助解决这些问题。专家可以用他们的实战经验来帮助项目。理想情况下，他们应当经常出席会议或通过电话进行沟通。

配有自动化测试、配置管理和频繁集成功能的技术环境

频繁的集成和测试是必不可少的，它们可以帮助团队尽早发现问题（错误，bug等）。持续集成可以阻止问题扩大，因为这些问题在早期就得到了暴露与解决。

极限编程（XP）[45]

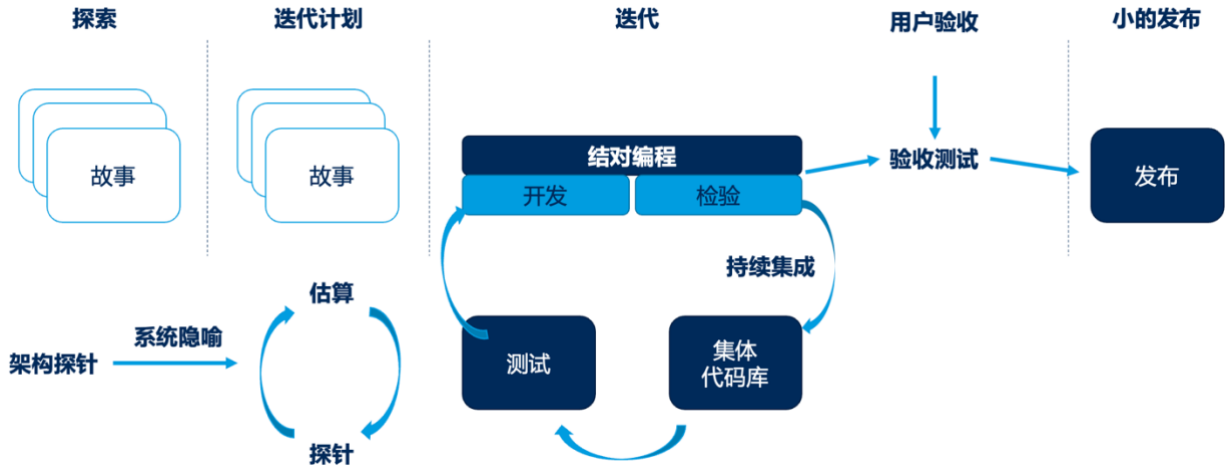
XP 是一种软件开发方法，它聚焦于客户的满意度。这保证了所开发的软件是基于客户的需要。实现以上内容的方法之一是将用户故事转化为有价值的商业资产。

极限编程方法的本质特征之一是持续和开放的交流。如果没有这一重要因素，这种实践就无法达到预期效果。这种沟通方式的一个重要收益是它增加了客户与项目团队的信心。

XP 是为了满足市场的需求而创建的，并以此适应服务供应商在开发新产品或服务时所面临的持续变化。它的成功可以归功于参与项目的各方之间的主要沟通与协作。这意味着客户、管理者、项目团队和任何其他相关人员从范围设定和故事被定义的那一刻起，直到测试和最终发布，他们都将会共同工作。

尽管大型团队正在使用 XP，并且也取得了成功，但该方法是为小型项目团队设计的。

图 35：极限编程（XP）一览



图片由 EXIN 基于以下信息提供：Meier, J. D. (2019). *Extreme Programming at a Glance*. https://jdmeier.com/extreme-programming-at-a-glance/

极限编程（XP）是如何运作的?

XP 的第一步是收集用户故事，并为那些可能存在风险的用户创建探针解决方案。

在初始步骤完成后，发布计划会议就要被排上日程。在这一阶段，应邀请所有有助于帮助完成发布计划的人。这些人可以包括客户、项目团队和管理者。这一阶段主要目标是是定义一个可以被大家所共同接受的目标（译者注：这里说的应该是发布计划内容，比如什么时候发布什么版本）。这个会议和目标设定之后是迭代计划会议，以便就即将而来的发布（的工作内容）达成共识。

只需上述几个步骤，项目团队已经向着开发所需特性的方向前进了。

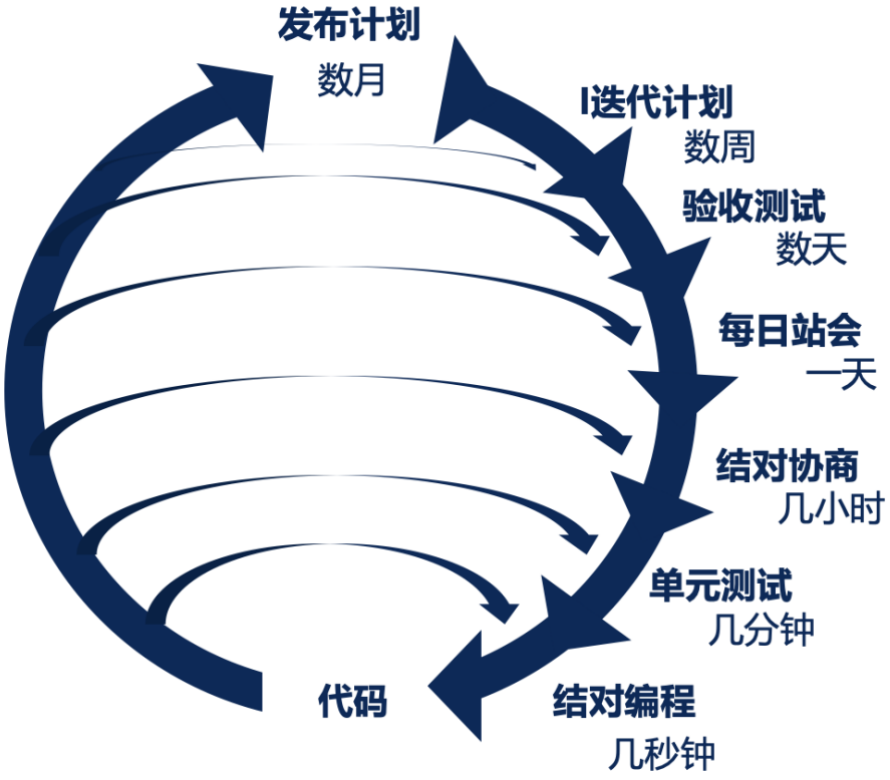
---

[45] 免费阅读 Wells, D. (2013). Extreme Programming: A gentle introduction. http://www.extremeprogramming.org/ and associated pages.

极限编程的规则

XP 依赖于频繁的反馈循环，如下图所示。

图 36：XP 中的计划回路



图片由 EXIN 基于以下内容创建：Wells, D. (2000). *Introducing Extreme Programming.* http://www.extremeprogramming.org/introduction.html

尽管以今天的角度来看，其中一些反馈循环似乎很长，但在 20 世纪 90 年代，它们是相当具有革命性的。

XP 的核心是规则理念。简而言之，XP 的规则是关于：

- 计划
- 管理
- 设计
- 编码
- 测试

计划

- 编写用户故事
- 通过发布计划会来创建发布时间表
- 频繁地小规模发布
- 项目被划分为多个迭代
- 冲刺规划从每次迭代开始

管理

- 为团队提供一个专用的开放式办公区
- 设定一个可持续的工作节奏
- 每天一次站立会议
- 可度量的项目速率
- 成员轮岗
- 当极限编程被破坏时要进行修复

设计

- 简单化
- 选择一个系统隐喻
- 在设计环节使用 CRC 卡（class responsibility collaborator，Class - 类名、Responsibilities - 类的职责、Collaboration - 类的协作关系）
- 创建探针解决方案以降低风险
- 不做过早的功能交付
- 随时随地重构

编码

- 真实客户参与
- 代码必须按照商定的标准编写
- 首先编写单元测试代码
- 所有发布的代码应使用结对编程
- 同步构建
- 频繁集成
- 配置专用集成计算机
- 代码集体所有权

测试

- 所有代码都必须进行单元测试
- 所有代码发布前必须要通过所有单元测试
- 当发现一个 bug 时，就立即为这个 bug 增加测试
- 经常进行验收测试，并公布测试结果

结对编程

结对编程是一种源自极限编程（XP）的敏捷技术，在极限编程中，两位 Developers 人员在一台计算机上协同工作。两人共同设计、编码和测试用户故事。理想情况下，两人技能熟练度相同且使用键盘的时间也相同的，这也是高级程序员教授初级程序员相关技能的有效方法。

动态系统开发方法 DSDM[46]

动态系统开发方法（DSDM）是一个聚焦于整个项目生命周期的框架。尽管许多人依然将 DSDM 当作一个组织名称，但该组织的新名称叫做敏捷商业联盟（Agile Business Consortium）。

该方法要求在每个步骤中只完成最低限度的工作，从而可以尽快处理下一项工作。这种做法背后的思维逻辑是"持续的变化对于一个项目而言再自然不过了"。众所周知，商业需求瞬息万变，因此只完成的所需步骤的必要工作，从而节省项目团队的精力、资源和时间。

DSDM 框架可以用于组织中不同的应用，从新产品或服务的开发到财务部门都是适用的。业务中的任何部分均可从中受益。

DSDM 的原则

1. 积极的用户参与是必要的
2. 必须授予 DSDM 团队制定决策的权力
3. 注重产品的频繁交付
4. 符合商业需求是接受交付物的必要标准
5. 迭代和增量式开发对于得到准确的商业解决方案是必不可少的
6. 开发过程的所有变化可逆
7. 在高层级上制定需求的基线
8. 测试自始至终贯穿于开发周期之中
9. 所有项目干系人的通力合作方式是不可或缺的

来源：敏捷商业联盟

DSDM 框架阶段

DSDM 框架包含 6 个阶段。包括前项目阶段与后项目阶段，项目阶段主要包括四个阶段：可行性研究 Feasibility）、项目信息了解（Foundations）、演进式开发（Evolutionary Development）和部署（Deployment）。

阶段1. 前项目阶段
阶段2. 可行性研究阶段
阶段3. 项目信息了解
阶段4. 演进式开发
阶段5. 部署
阶段6. 后项目阶段

阶段 1 - 前项目阶段

明确定义目标是项目准备工作的一部分。前项目阶段有助于确保正确的建立项目，同时仅启动正确的项目。

[46] 免费阅读: Agile Business Consortium (2014). The DSDM Agile Project Framework (2014 Onwards). https://www.agilebusiness.org/page/TheDSDMAgileProjectFramework

阶段 2 - 可行性研究阶段

正如这一阶段名称所描述的，这一部分重点是，从技术和成本效益的角度来评估所讨论的项目是否具备可行性。该阶段不应占用太多时间，仅需足够时间用于研究并确定项目是否值得实施或是否具有可行性。

阶段 3 - 项目信息了解

此阶段的目标是了解工作范围。这其中包括如何、何人、何时以及在何处开展实际工作。项目生命周期仍取决于如何应用 DSDM 流程。

阶段 4 -演进式开发

解决方案开发团队将应用包括迭代开发，MoSCoW 或时间盒在内的实践来改进解决方案。其目的是进一步演进解决方案，使其准确无误、从技术角度来看是内置的"正确方式"、并满足商业需要。随着项目进展，团队将持续迭代开发和测试解决方案。

阶段 5 - 部署

此阶段侧重于三项主要活动：整合、评审和部署。该阶段旨在发布最终方案或其子集。项目将在最后一次发布后正式关闭。

阶段 6 - 后项目阶段

后项目阶段将通过效益评估的方式检查商业效益预期的实现情况。

LeSS

LeSS 代表大规模 Scrum。该方法适用于多个团队同时合作开发同一个产品或服务。

LeSS 可应用于以下情况：

- 一个特定的开发需要多个团队
- 多个团队基于一个共同目标共事
- 多个团队协同开发一个产品或服务

如果您的项目不符合这些要求，那么最好使用标准 Scrum。LeSS 的 10 个原则。

LeSS 的十大原则

LeSS 的十大原则总结在下图中。

图 37：LeSS 中使用的理念

图片由 EXIN 基于以下内容创建：The LeSS Company B.V. (2014). *Large Scale Scrum is Scrum [Visual]*. https://less.works/img/principles/principles.pdf.

大规模 Scrum 依然是 Scrum

LeSS 仍是 Scrum 的一种形式。然而，它被扩展用于大型产品或服务开发。LeSS 提供了一套与指南相结合的规则，用于在多团队情境中应用 Scrum。但它并不是一个新的 Scrum 框架。

透明

主要成果必须对各方可见。创造较短时间内的反馈循环可以提升透明度。而透明度对于确保自适应控制和改进是必要的。

少即是多

该原则是大规模 Scrum（LeSS）的核心，承认复杂的组织解决方案是不利的，可能会顾此失彼。它旨在通过采用更简单与不同的问题解决方案来消除复杂性。

关注整个产品

在 LeSS 中，重点是完整的产品。因为产品的单个贡献部分直到被组合在一起成为最终产品时才具有实际价值。毕竟，客户购买的是整个产品，而不是其中一部分。保持对"整体"的关注是大型开发团队面临的最大挑战之一。

以用户为中心

与关注产品一样，大规模的 LeSS 也给客户满意度带来了风险。在 LeSS 中，产品负责人是客户/用户和项目团队之间沟通的桥梁。为了确保以用户为中心，团队是基于端到端的特性和组件来组织的。

持续改进，力求完美

作为精益思想的两大支柱之一，"持续改进，力求完美"也是 LeSS 的一部分。在 LeSS 中，变革是可预期的、持续的，并且被接受的。

精益思想

尊重人和持续改进是 LeSS 采用的精益生产的核心思想。

系统思维

该原则表示长期的、系统化的改进更优于快速解决方案。

经验性过程控制

该 Scrum 概念被应用在 LeSS 中，以确保团队拥有"恰到好处的流程"。通过这种方式，他们在一个透明、检视和适应的循环中工作。

队列理论

这个关于项目如何在系统队列中移动的理论是一种思维工具，可以用来提高处理大批量工作的能力。这与大规模 Scrum 尤其相关。

大规模敏捷框架（SAFe®）[47]

SAFe 是一个免费可用的知识体系，它是为了解决在多团队之间扩展敏捷时遇到的问题而创建的。这种企业级开发方法结合了精益和敏捷的原则。框架中的组织和工作流模式旨在支持组织扩展其精益和敏捷实践。

9 个 SAFe 精益敏捷原则

1. 采用经济视角
2. 应用系统思维
3. 假设可变性；预留方案
4. 以快速整合的学习周期进行增量式构建
5. 基于对可工作系统的客观评估设立里程碑
6. 可视化并限制在制品（WiP），减少批次规模，管理队列长度
7. 应用节奏，通过跨领域规划进行同步
8. 释放知识工作者的内在动力
9. 去中心化的决策

SAFe 框架

SAFe 网站提供了关于 SAFe 的简要说明。在 SAFe 中，有四种配置：

- Essential
- Portfolio
- Large solution
- Full

Essential

Essential SAFe 是最基本配置。它描述了所需最关键的元素，并旨在提供框架最大的益处。该配置涵盖团队和项目集层级的待办事项列表，项目集层级内容也被称为敏捷发布火车或 ARTs。

Portfolio

Portfolio SAFe 包含对战略方向、投资资金和精益治理的关注。

Large solution

Large Solution SAFe 允许多个计划的协调和同步，但不考虑投资组合。在 SAFe 的早期版本中，该层级也被称为价值流。

Full

Full SAFe 包含了其他三个层级的所有内容。

---

[47] 免费阅读： Scaled Agile (2021). Scaled Agile Framework. https://www.scaledagileframework.com/

规范敏捷交付

规范敏捷交付（DAD）是一个框架，它提供了适合企业需求的特定环境下的指南，以帮助企业快速生产高质量的产品。它是一个混合模型，由各种世界公认的精益敏捷方法集合在一起形成，如 Scrum、Kanban、极限编程（XP）、敏捷建模、统一过程等等。

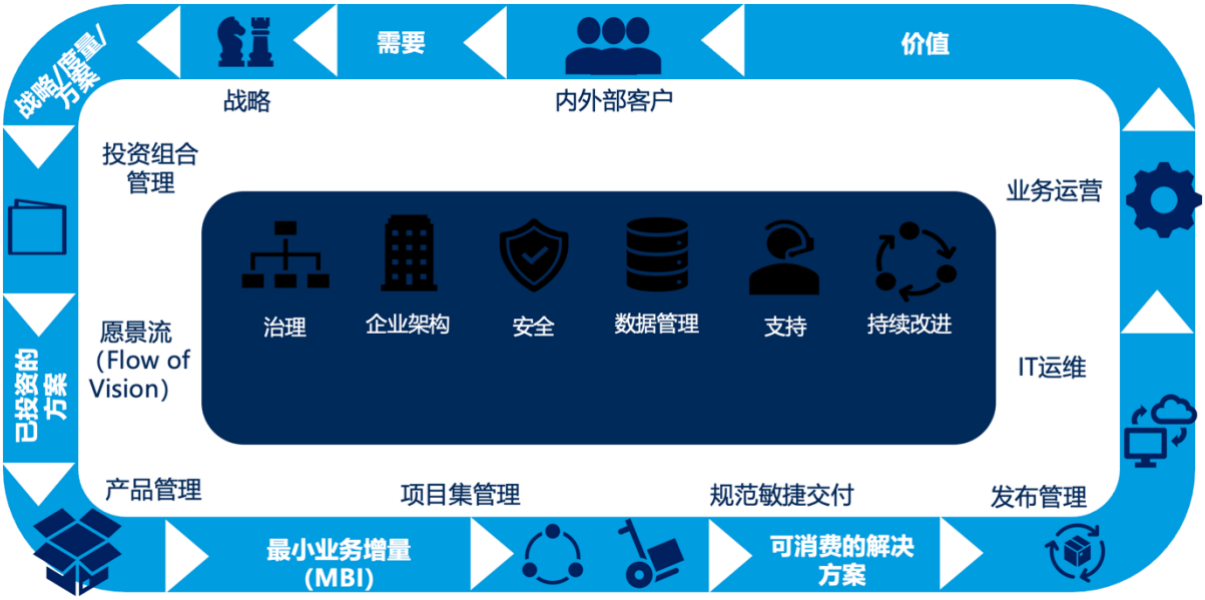
该框架最初是由 IBM Rational 于 2009 年初至 2012 年 6 月所研发。IBM 团队由 Scott Ambler 领导，与包括 Mark Lines 在内的商业伙伴密切合作。

2012 年 10 月，DAD 知识产权的所有权有效地移交给了规范敏捷联盟（Disciplined Agile Consortium），这一事实在 2014 年 6 月得到了 IBM 的法务认可。

DAD 后来发展成一个全面的工具包，现在被称为规范敏捷 5.x，于 2020 年 5 月发布，并发表了一本名为*《选择你的"哇！"时刻：规范敏捷交付手册优化你的工作方式》*（Ambler & Lines, 2020）的书。

DAD 目前只是规范敏捷的一个子集，规范敏捷涵盖了更广泛的范围。

图 38：各种角色如何在 DA 框架中协同工作



图片由 EXIN 基于以下内容创建：Project Management Institute, Inc. (2021). *Introduction to Disciplined Agile® (DA™)*. https://www.pmi.org/disciplined-Agile/introduction-to-disciplined-Agile.

看板

看板是流行于制造业中的一种精益技术，最近一段时间内它在特定的 IT 项目和运营中得到了越来越多的应用。在 IT 环境中，人们所说的“看板”通常是一种开发方法，它是各种 Scrum 元素与看板技术的结合。

“看板在 Scrum 中的应用”在正文使用看板方法中已经进行了详细描述，此处不再赘述。

附录 B - 关于发布与敏捷环境中常用的发布管理工具的更多信息

发布还是不发布，这是个问题

实践者对在 Scrum 中使用发布概念的批评越来越多，并且言之成理。发布概念是对没有预先规划的嘲弄，而同时对在 Scrum 中发布的支持者会告知你这不是详细的规划！好吧，事实的确如此。

注意：自 2011 年起，发布的概念就不再是 Scrum 指南的正式内容了。事实上，为什么 Scrum 指南中删除了发布计划？答案简单明了：不使用发布计划也可以成功应用 Scrum。

不需要发布计划"可能会成为产品健康度的一个正向指标。Scrum 旨在通过短迭代不断向客户交付价值。使用发布计划可能会延迟价值交付、客户反馈与投资回报（RoI）。凭借成熟的开发实践和卓越的推广策略，可以在每个冲刺中交付价值[48]。

让我们消除发布管理的支持者提出的一些谬论，并借此探究为什么传统的发布管理方法在 Scrum 中不是一个好主意。这些谬论包括：

谬论 1

在复杂的和可扩展的环境中，预先规划是必要的。

现实 1

是的 - 但是所有需要的预先规划都可以通过确保产品待办事项列表得到适当梳理来完成。

谬论 2

产品待办事项列表的梳理是不足以规划到同时兼顾协调性、依赖关系和顺序。

现实 2

是的-但是使用像 Nexus 这样的可扩展机制就足以确保作为 Nexus 一部分的 Scrum 团队之间的适当协调，且足以正确处理依赖关系，即使这些依赖关系是跨迭代的。

[48] Scrum.org (2021). Gone Are Release Planning and the Release Turndown. https://www.scrum.org/resources/gone-are-Release-planning-and-Release-turndown

谬论 3

像 Nexus 这样的方法不足以满足连续的冲刺，因此也不能满足随时间变化的依赖性。

现实 3A

像人们在发布计划中所做的那样，对连续的冲刺中将要发生的事情进行详细的规划，是在把 Scrum 重新变成瀑布（Waterfall）。要理解顺序、依赖关系以及下一个冲刺中最可能发生的事情，只需要适当管理你的产品待办事项列表就足够了。除此之外，像 Nexus 这样的可扩展方式会专门记录和管理冲刺之间的依赖关系与顺序，但也仅此而已，因为其他的（依赖关系与顺序）都应在管理产品待办事项列表时处理。

现实 3B

发布方法最不利的后果之一是，团队会陷入诸如"在冲刺 2 中，我们会做 x，这已经决定了"的陷阱。这违背了敏捷原则 2 ，并可能也违背了敏捷原则 1、4、5、10 和 11。

谬论 4

由于业务需要结束日期，所以我们使用发布的概念，从而创建了可预测性，进而帮助我们实现业务目标。

现实 4

这是 WaterScrumFall（瀑布式 Scrum），这并不是敏捷。

关于"发布"的指南

然而，如果你认为你的组织应持续使用发布管理，那么这里有一些指导意见供参考。

- 不要将发布周期延长到超过三个冲刺。
- 确保所有人都知道发布计划可能且应该在每次迭代后发生改变，除非业务中的一切都保持不变，而这是极不可能的。
- 创建和使用发布待办事项列表（与冲刺待办事项列表类似，但是对于一次发布中包含的所有冲刺）必须是高层级的且极具灵活性。这意味着新的事项将不断地进入发布待办事项列表。在没有更多的团队加入发布的情况下，部分发布待办事项列表中的工作项需要被剔除。
- 使用发布计划作为一种指导和方法来专注于产品目标；你会发现一些必须快速处理的未知事情。
- 灵活应对且专注于为客户做正确的事情，而不是墨守陈规地坚持计划。
- 要注意产品待办事项列表中的依赖关系与顺序。
- 使用像 Nexus 这样的可扩展机制。如果你这么做，你的发布计划和管理将更为简单和高层级。注意：像 Scrum@Scale 这样的方法包含了发布管理的理念，如对其探源溯流，或将受益良多。
- 在发布计划中不要做任务分解；把其留至每个冲刺周期。从本质上来说，这意味着，如果你不使用像 Nexus 这样的可扩展机制，那么你将在每个冲刺周期中都进行一些发布计划。
- 不要最终走向"WaterScrumFall"（瀑布式 Scrum）。

燃尽与估算

可以将发布燃尽图视作一个特别有用的追踪发布的工具。这个想法与普通的燃尽图并无二致，只需在过去的方法上稍加调整。

Scrum 项目的进展可以通过发布燃尽图来追踪。管理发布的 SM 应在每个冲刺结束时更新发布燃尽图。

像冲刺燃尽图一样，纵轴显示每个冲刺开始时的剩余工作量，为了表达这一点，可以使用故事点数、理想人天、团队天数、或你在组织中选择的任何参照物。横轴显示时间，但在本例中，显示的是发布（项目）完成前的冲刺次数。如果一个发布不超过三个冲刺，为什么还要大费周章呢?

这是个好问题。

如果你使用发布管理，那么很可能是因为你轻信了上面所说的谬论 4。

如果工作不断地被添加，并且你的客户不希望降低发布事项列表中的产品待办事项的优先级时（降低优先级，本质上就是允许你移除事项并将它们放回产品待办事项列表中），任何人都知道这将花费更久的时间。

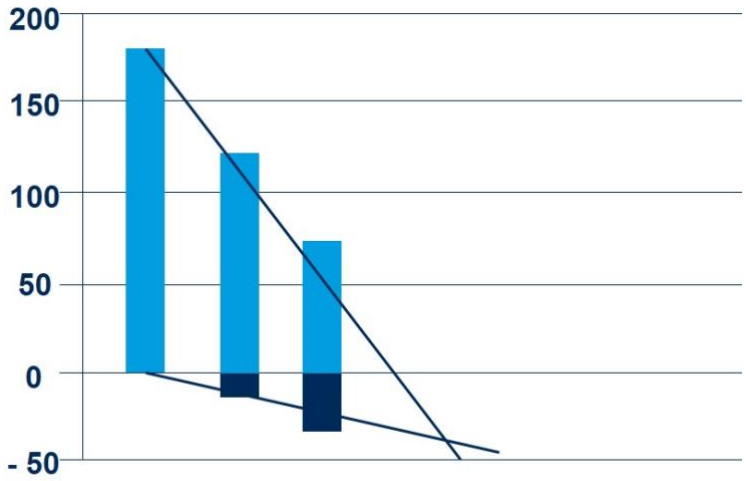
但是更久是多久?

发布燃尽图会帮助你找到答案。

发布燃尽图不仅可以降为零，还可以让你在纵轴上降为负值。随着发布期间添加额外的工作，在每个冲刺周期中添加的工作，都会被添加到零线以下。

为什么说这有所助益呢？因为通过这样做，你可以使用两条趋势线来预测完成发布还剩多少个冲刺。

图 39：发布燃尽条



图片由 EXIN 基于以下内容创建：Botha, J. (2018). *Scrum Masters and Product Owners [Courseware]*. GetITright.

发布燃尽图有两条趋势线：一条是零线以上的数据；另一条是零线以下的数据。交叉点会告知负责指导发布的 SM：为了应对冲刺 2 和冲刺 3 中新增工作，需要一个额外的冲刺周期来完成发布。根据上述三个冲刺信息推导可知，进度情况并不理想。

实际情况是，这通常看起来会更糟糕，因为预计 10 个或更多的冲刺的情况并不少见。而这其实是一个瀑布式（Waterfall）项目。

为发布计划使用甘特图

一种常见的做法是使用甘特图来直观地提醒发布中的团队在哪一个冲刺中，发布待办事项列表会发生什么样的成果。它还用于显示产品待办事项列表中的待办项及其相关任务之间的交叉依赖。

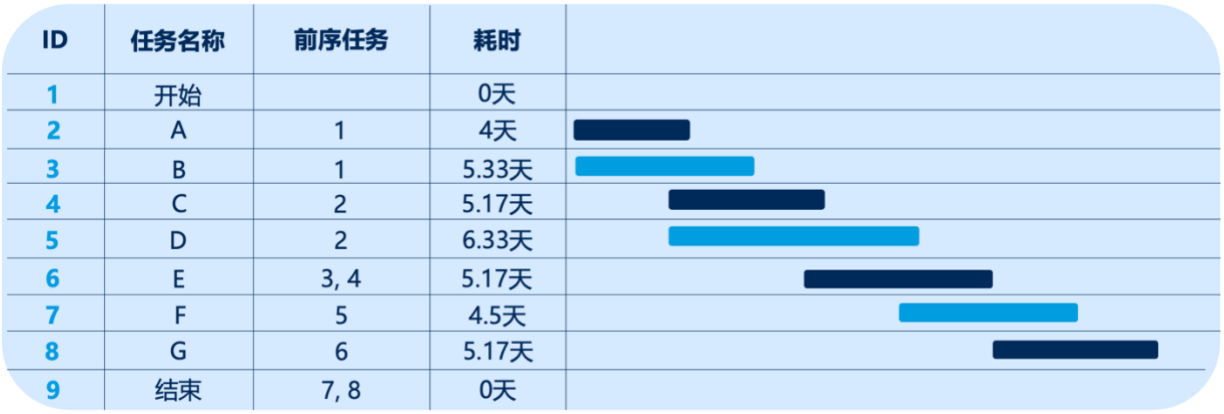
但是，什么是甘特图呢?

甘特图是表示随时间推移计划任务、任务之间的关联以及他们之间的依赖关系的可视化图表。

这个概念是由 Henry L. Gantt 在 1910 年左右提出的，是在他与 Frederick Taylor 一起工作时，用于描述工厂和车间的生产规划与资源装载的一种方法。甘特图最初的主要应用之一是美国在第一次世界大战期间为战争规划物流方案。

你们都在项目会议上看过这些图表，它们看起来像这样：

图 40：使用甘特图展示的发布计划



ID	任务名称	前序任务	耗时
1	开始		0天
2	A	1	4天
3	B	1	5.33天
4	C	2	5.17天
5	D	2	6.33天
6	E	3, 4	5.17天
7	F	5	4.5天
8	G	6	5.17天
9	结束	7, 8	0天

图片由 EXIN 由 EXIN 基于以下信息创建：Gantt Chart. (2021).2021 年 2 月 14 摘取自 https://en.wikipedia.org/wiki/Gantt_chart.

原则上来说，使用甘特图作为可视化数据的手段没有错。但更重要的是它的使用情景和所暗含的内容。如果将其用于详细的前期规划，这将是对敏捷和 Scrum 的无情嘲弄。

书目与参考资料

Agile Business Consortium (2014). The DSDM Agile Project Framework (2014 Onwards). https://www.agilebusiness.org/page/TheDSDMAgileProjectFramework

Ambler, S. & Lines, M. (2020). Choose your WoW!: A Disciplined Agile Delivery Handbook for Optimizing Your Way of Working (WoW). Project Management Institute.

Bigelow, S. J. (2020). 7 techniques for better Agile requirements gathering. https://searchsoftwarequality.techtarget.com/tip/7-techniques-for-better-Agile-requirements-gathering

Botha, J. (2018). Scrum Masters and Product Owners [Courseware]. GetITright.

Botha, J. (2019). Agile: A Manager's Guide to Unlocking Business Value. Amazon Digital Services LLC - Kdp Print Us.

Botha, J. (2020). Scrum Lego-game workbook [Courseware]. GetITright.

Botha, J. (2021). [Courseware]. GetITright. Retrieved in personal communication from the author of the book.

Botha, J. (2021). Lean fundamentals for IT [Courseware]. GetITright.

Cohn, M. (2005). Agile Estimating and Planning. Pearson Education.

Cohn, M. (2009). Four Attributes of the Ideal Pilot Project. https://www.mountaingoatsoftware.com/blog/four-attributes-of-the-ideal-pilot-project

Cohn, M. (2010). Succeeding with Agile. Addison-Wesley.

Crystal Methods. (2021). Retrieved on September 23, 2021 from

EXIN Holding B.V. (2019). Agile Methodologies. EXIN Holding B.V. https://dam.exin.com/api/&request=asset.permadownload&id=3144&type=this&token=8659eabedff466d86ac2eba5ed72b33d

Exoskeleton. (2021). Retrieved on September 23, 2021 from https://en.wikipedia.org/wiki/Exoskeleton

Gantt Chart. (2021). Retrieved on September 23, 2021 from https://en.wikipedia.org/wiki/Gantt_chart

Goldratt, E. M. & Cox, J. (2012). The Goal. (3rd Edition) North River Press.

Hiatt, J. M. & Creasey, T. J. (2012). Change Management: The People Side of Change. Prosci Learning Center Publications.

https://en.wikiversity.org/wiki/Crystal_Methods

Kano+ (2020). The Kano model – Assessing Product Features based on Customer Satisfaction. Kano+ https://kano.plus/about-kano#analyze-a-study

Maher, R., & Kong, P. (2016). Cross-Team Refinement in Nexus™. Scrum.org. https://www.scrum.org/resources/cross-team-refinement-nexus

Maher, R., & Kong, P. (2016). The Nexus Integration Team. Scrum.org. https://www.scrum.org/resources/nexus-integration-team

Maher, R., & Kong, P. (2016). Visualizing the Nexus Sprint Backlog. Scrum.org. https://www.scrum.org/resources/visualizing-nexus-sprint-backlog

Martin, B. A. (1980). The goal: to improve credibility in the reporting of engineering progress. Project Management Quarterly, 11(2), 14–22.

McKinsey & Company (2019). The journey to an Agile organization. https://www.mckinsey.com/business-functions/organization/our-insights/the-journey-to-an-agile-organization
Meier, J. D. (2019). Extreme Programming at a Glance. https://jdmeier.com/extreme-programming-at-a-glance/
Overeem, B. (2016). The 11 Advantages of Using a Sprint Goal. https://www.scrum.org/resources/blog/11-advantages-using-sprint-goal
Pichler, R. (2010). Agile Product Management with Scrum. Addison-Wesley.
Project Management Institute, Inc. (2021). Introduction to Disciplined Agile® (DA™). https://www.pmi.org/disciplined-Agile/introduction-to-disciplined-Agile
Prosci (2021). Best Practices in change management benchmarking report. https://www.prosci.com/resources/articles/change-management-best-practices
Rad, N. K. (2021). Agile Scrum Handbook (3de ed.). Van Haren Publishing.
Rother, M. (2018). The Toyota Kata Practice Guide: Practicing Scientific Thinking Skills for Superior Results in 20 Minutes a Day. Mcgraw-Hill Education.
Scaled Agile (2021). Scaled Agile Framework. https://www.scaledagileframework.com/
Schwaber, K., & Scrum.org (2021). The Nexus™ Guide – The Definitive Guide to Scaling Scrum with Nexus. Scrum.org. https://www.scrum.org/resources/nexus-guide
Schwaber, K., & Sutherland, J. (2020). The Scrum Guide – The Definitive Guide to Scrum: The Rules of the Game. Scrumguides.org. https://scrumguides.org/scrum-guide.html
Scrum.org (2021). Gone Are Release Planning and the Release Turndown. https://www.scrum.org/resources/gone-are-Release-planning-and-Release-turndown
Scrum.org., Vacanti, D., & Yeret, Y. (2021). The Kanban Guide for Scrum Teams. Scrum.org. https://www.scrum.org/resources/kanban-guide-scrum-teams
Senge, P. (2006). The Fifth Discipline. Bantam Doubleday Dell Publishing Group Inc.
Sutherland, J., & Scrum Inc. (2021). The Scrum@Scale® Guide – The Definitive Guide to Scrum@Scale: Scaling that Works. Scrum.org. https://www.scrumatscale.com/wp-content/uploads/2020/12/official-scrum-at-scale-guide.pdf
Technical Debt. (2021). Retrieved on September 23, 2021 from https://en.wikipedia.org/wiki/Technical_debt
The LeSS Company B.V. (2014). Large Scale Scrum is Scrum [Visual]. https://less.works/img/principles/principles.pdf
Underwood, J. (2013). Musselwhite and Ingram's Change Style Indicator. https://innovategov.org/2013/08/15/musslewhite-and-ingrams-change-style-indicator/

Wells, D. (2000). Introducing Extreme Programming.
http://www.extremeprogramming.org/introduction.html
Wells, D. (2013). Extreme Programming: A gentle introduction.
http://www.extremeprogramming.org/

联系 EXIN

www.exinchina.cn

info.china@exin.com

WeChat ID: EXINCH