Don Reinertsen 推荐序

PEARSON

Agile Software Requirements

Lean Requirements Practices for Teams, Programs, and the Enterprise

敏捷软件需求

团队、项目群与企业级的精益需求实践

Dean Leffingwell 著

刘磊 傅庆冬 李建昊 雷迅 译

清华大学出版社

Agile Software Requirements

Lean Requirements Practices for Teams, Programs, and the Enterprise

敏捷软件需求

团队、项目群与企业级的精益需求实践

Dean Leffingwell 著

刘磊 傅庆冬 李建昊 雷迅 译

清华大学出版社

北京

内 容 简 介

本书全面介绍了如何在敏捷环境中管理软件需求，全书共四部分 24 章。第 I 部分提出企业的敏捷需求全景图，针对项目团队、项目集和项目组合这三个级别，描述了一个整体的敏捷需求过程模型。第 II 部分描述一个简单的、轻量级的但同时又宽泛的模型，敏捷项目团队可以使用这个模型来管理需求。第 III 部分展示如何为需要多团队合作的复杂项目确定敏捷需求。第 IV 部分指导企业如何为大型的“专用系统”、应用套件和产品项目组合制定敏捷需求。

本书适合软件开发人员、测试人员、执行主管，项目/项目集经理、架构师和团队领导阅读和参考，可以帮助他们在敏捷转型过程中去除障碍，打造出优秀的软件产品。

Simplified Chinese edition copyright © 2014 by **PEARSON EDUCATION ASIA LIMITED and TSINGHUA UNIVERSITY PRESS.**
Original English language title from Proprietor's edition of the Work.
Original English language title: Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise
EISBN: 978-0-321-63584-6
All Rights Reserved.
Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Pearson Education.
This edition is authorized for sale only in the People's Republic of China (excluding the Special Administrative Region of Hong Kong and Macao).

本书中文简体翻译版由 Pearson Education 授权给清华大学出版社在中国境内(不包括中国香港、澳门特别行政区)出版发行。

北京市版权局著作权合同登记号　图字：01-2011-5339
本书封面贴有 Pearson Education(培生教育出版集团)激光防伪标签，无标签者不得销售。
版权所有，侵权必究。侵权举报电话：010-62782989　13701121933

图书在版编目(CIP)数据

敏捷软件需求：团队、项目群与企业级的精益需求实践/(美)莱芬韦尔(Leffingwell, D.)著；刘磊等译. --北京：清华大学出版社，2015

书名原文：Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise

ISBN 978-7-302-35447-5

Ⅰ. ①敏… Ⅱ. ①莱…②刘… Ⅲ. ①软件开发—项目管理—研究 Ⅳ. ①TP311.52

中国版本图书馆 CIP 数据核字(2014)第 023043 号

责任编辑：文开琪
封面设计：杨玉兰
责任校对：周剑云
责任印制：李红英
出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：清华大学印刷厂
装 订 者：三河市溧源装订厂
经　　销：全国新华书店
开　　本：185mm×230mm　**印　张**：34.25　**插　页**：2　**字　数**：579 千字
版　　次：2015 年 1 月第 1 版　　**印　次**：2015 年 1 月第1 次印刷
印　　数：1～3500
定　　价：79.00 元

产品编号：044252-01

推 荐 序

（Don Reinertesn）

为什么一些产品开发项目达不到预定的经济目标呢？研究表明，80%～85%的项目失败归咎于需求不正确。有经验的开发者知道，管理软件需求是比技术实施更严峻的挑战。而且，虽然数十年来我们一直都知道这一点，但真的在这方面没有任何改善。为什么呢？起初，我们只按职能来组织开发人员，出了问题就推诿到工程之外的部门，怪市场和生产管理部门。后来，随着跨职能团队的采用，我们要求团队要倾听客户的声音，并假定这样做可以解决需求不当这个问题。

问题并没有得以解决。我们从来没有质疑过先行开发实际需求这一思路，只是告诉人们要更努力尝试。有一个事实我们视而不见：许多客户并不知道自己想要什么。有一个事实我们视而不见：即使客户知道自己想要什么，也无法清楚描述。有一个事实我们视而不见：即使客户能够描述自己的需求，但描述的也通常是提议方案，而不是他们真正的需求。比如，客户告诉我们，他们想要便于携带的旅行箱，并且告诉我们不能太重。我们这样做了，他们却拒绝我们优雅的设计，要买竞争对手更笨重的设计，即带有轮子的行李箱！

一个悲哀的事实是，根本没有“客户的声音”，有的只是一些表达不同需求的杂音罢了。即使针对一个客户，我们也要平衡技术决策者、最终用户、系统管理员和财务决策者的要求。所有这些角色对不同属性有着不同的权衡，同时随着产品使用体验的增多，他们会改变自己的权重。与此同时，我们还需要了解分销商、监管者、制造和现场服务的需求。如果我们只关注用户，就会错过 Dean 所说的“非功能需求”。

这个问题是动态而不是静态的。在开发工作进行过程中，环境不断在变，竞争对手引入新的产品，客户需求发生演变。如果在开始设计之前开发实际需求工件不可行，有什么替代选项呢？在我看来，我们应该开始相信，即使是最适当的

需求，也包含大问题，而且这种问题会随着时间的推移不断加剧。这想法改变了我们的关注点，我们不再相信自己听到的是来自客户的高精确信号，而必须承认它是一种嘈杂、低精确度的信号，是必须持续检查错误的信号。我们不再以大量前期投入来产生完美的需求，而是投资建立足以用来快速检测与纠正的过程和基础设施，利用它们来快速检测和纠正处于解决方案和客户变动需求之间的半成品。

要想证明这一替代方案，开发大型系统就是最好的测试。在小项目上行之有效的很多方法，遇到大项目就会分崩离析。例如，在小系统中，成本和收益通常是局部的，团队做出的局部优化决策时，不会损坏系统性能。但是，这对大系统不成立，在大型系统中，我们必须处理地理、组织方面的、临时发生的经济影响。

我们需要更好的方法来理解和管理软件需求。Dean 在这本书中提供了这些方法。他从三个非常有用的智慧宝库提炼思想：传统的管理做法、敏捷方法和精益生产开发。通过综合这三种方法的优点，他提供的方法比任何一种单一方法更奏效。

首先，虽然这样说可能不时髦，但传统管理实践仍然为我们提供了一些非常有用的方法。我们的前辈并不都是愚蠢的傻瓜，找不出可行的方案。过去几十年来，我也见证过一些简单的技术和生产路线图产生很好的结果，它们可以保证技术工作尽早开始，以免偏离关键路径。它们在技术工作及其服务的项目之间建立起较强的逻辑关联。虽然不必盲目接受所有的传统实践，但丢弃前辈总结得出的所有智慧也是不明智的，Dean 为我们展现了如何在项目集和项目组合层级应用这些卓越的思想。

其次，敏捷社区已经发展出一套非常强大的思想体系，并已经产生了令人瞩目的成果。这些方法的快速成长有一个很好的理由，因为它们有用。敏捷把大批量瀑布模型分解到一系列有时间盒限制的迭代中。这些小批量的迭代大大加快了反馈，产生巨大的效益。

由于很多敏捷成功都是发生在小项目中的，所以人们自然想知道它是否同样

试读结束：需要全本请在线购买： www.ertongbook.com

适用于大型系统。虽然我非常尊重敏捷方法的价值观，但我更赞同 Dean 的观点，认为这些方法必须加以扩展以满足大型系统开发的需要。如果认为大型系统的架构也会顺其自然地浮现，任何不足都可以通过重构来消除，这是很有风险的。例如，依照设计方案，一艘海军战舰的寿命为 30 年。优秀的海军建造师会预先考虑威胁的演变、新兴技术和使命的改变，所以我们在建构这样的系统时就不能“坐等”架构浮现。只要意识到系统层级的独特管理挑战，我们就可以着手投资于组织性基础设施以应对这个挑战。Dean 在书中展示了如何运用“架构跑道”（architectural runway）这样的敏捷方法扩展来做到这一点。

Dean 还借鉴了我所称的“第二代精益生产开发”的思想。在产品开发中运用精益思想的最初大部分尝试，都专注于工作标准化和可变性降低这样的思路。它们缺乏敏捷的本源，即开发卓越的新方案要求我们在不确定的环境中不断学习。这些精益生产开发方法现在已经演化，成效令人瞩目。例如，现在 “看板”方法的作用是限制在制品、加快反馈和使过程对所有参与者可见。从本书中，可以看出这些思想在项目集和项目组合层级对 Dean 方法论的影响。Dean 也认识到当下强调经济效益的重要性。这种强调有助于我们做出更好的抉择，使我们能够用管理层容易理解的词汇向他们解释为什么要这样选。

在阅读本书过程中，我建议大家关注以下几点。首先，试着理解为什么这些特定方法有效，而不是只停留于理解方法本身。如果理解幕后的工作机制，就可以更容易地针对自己的独特环境加以调整。其次，把这些思想作为一套有用的模式包，而不是一些必须严格成组采纳的实践。这样做的好处是，周期短，抵制少，见效快。最后，在应用这些思想的过程中，要努力保持平衡。很多人会出于本能，倾向于某些特定的思想，因为它们解决了他们觉得重要的问题。其他有一些领域，也许长期不受关注，但事实上，这些不起眼的地方往往蕴藏着很不错的商机。

中 文 版 序

（Dean Leffingwell　李建昊）

企业的数字化转型意味着软件开发人员必须构建更大、更好的软件系统，而软件开发也成为当今世界最具挑战性的智力活动之一。为了应对这一挑战，许多思想领袖已经开始采用众多的新兴知识，比如，敏捷方法、精益和系统思维、产品开发流程原理等。在本书中，我尝试着整合这些互补的方法和原则，使其成为一个可以为企业中的团队、项目集和项目组合等各个级别提供指导的全面完整的框架。

我非常感谢李建昊（James Li）和本书的其他译者，以及清华大学出版社文开琪老师，促成了本书中文版的问世，从而为中国广大的软件人员，在持续改进和提升企业级软件系统构建方法和实践中，在不断分享经验和教训等方面提供帮助。在本书中，我描述了如何通过定义和管理软件需求的机制来定义和管理价值流。更重要的是，可以通过价值流帮助我们获得快速反馈，演进方案系统，从而确保最终满足用户的真正需求。

如果你已经对这些概念非常熟悉，这可能是因为中国已经开始进行了一些基础性的研究工作，在中国的许多大型企业或者许多跨国公司在中国的分支机构已经在开展这方面的工作了。

自本书首次出版以来，我们的工作仍在继续，我们在规模化敏捷框架（Scaled Agile Framework，SAFe）的环境中，进一步开发和扩展了本书所论述的原则，更多信息请访问 ScaledAgileFramework.com。SAFe 是一种基于知识的、行之有效的最佳实践，可以为你的企业和业务带来精益和敏捷开发的益处。SAFe 是公开的，可以自由使用，所以你可以立即在业务环境中加以使用。经过 Scaled Agile 公司及其全球合作伙伴提供的培训、认证、资料和咨询，SAFe 得以良好的推广和扩展，更多详情请参考 ScaledAgilePartners.com。

我们很高兴能够把这些思想带给至关重要的中国软件市场。然而，有一点值得注意，实施这些实践是一项非常艰苦的工作。必须具备新的价值观、新的工作方式以及愿意参与并得到充分授权的从业人员。其实，没有什么特殊的或隐藏的秘密可以解开这个“魔力问题”（即企业的规模化敏捷实施），如果说能找到一些“秘笈”的话，那么这些“秘笈”可能来自曾经受到组织环境问题困扰而思考并勇于承担变革的人。ASR（敏捷软件需求）和 SAFe 仅仅提供了模板，但真正的实施工作是要靠人来完成的。

我们将这部分的工作留给你们，亲爱的读者们。

让我们一起努力，因为更好的软件可以让世界变得更加美好！

译序 1

——浅谈大规模敏捷框架 SAFe

（傅庆冬）

这是 Dean Leffingwell 的第二部关于规模化敏捷的著作。从 2007 年出版《可伸缩敏捷开发》以来，Dean 在推动企业进行规模化敏捷转型方面的思想也得到进一步丰富和完善。这部著作就是过去 4 年在理论认识上的进展以及在一些大型企业或大型软件开发项目上积极实践的经验总结。

最早接触大规模敏捷框架 SAFe 是当时所在的公司正在考虑采用什么样的规模化敏捷实施方法论来进行组织的敏捷转型。所以初次接触这个模型的人（基本都是通过访问 SAFe 的网站）通常认为这是对大型企业实施敏捷的组织结构管理及流程改进的一套方法论，其目的就是通过调整组织结构来推进敏捷转型。近几年来，敏捷社区里很多人也乐于讨论如何帮助大型企业进行敏捷转型，很多话题都涉及组织结构调整，一个比较普遍的观念是，敏捷开发不再需要传统的层级式的管理，程序员文化才是发展前途，组织结构的扁平化才是出路。因此，当人们初次看到这个模型，很自然地觉得这是一个向传统管理模型投降的方法论，从全景图上就可以很清晰地看到一个三层管理结构，这通常与自己在敏捷实践或者对敏捷转型的期望不符。很多人在自己的博客或者在一些研讨会上指出，这个模型是失败的，不符合敏捷思想，是披着敏捷外衣的伪敏捷。这其实是一个误解。

SAFe 是一个关于探讨“敏捷”组织结构及管理方式的模型吗？不是的

本书给出一个大规模敏捷框架，即 SAFe 的全景图。SAFe 的全景图初看过于复杂，人物角色太多。人们倾向于在这样的示意图中先找自己的位置，然后再看在这个模型中相对应的位置定义的职责范围。比如，团队成员会找到自己的位置：在全景图的左下角，显示了开发团队的组成结构，包括程序员、测试、Scrum

Master 和产品负责人（Product Owner，PO），然后，这样的团队面对一个团队待办事项列表（team backlog），这是一个看上去比较典型的敏捷开发团队的组织结构。架构师则发现，在项目团队中没有他们的位置，在项目集（program，一组相互关联且被协调管理的项目，也译为“项目群”，本书后文统一为“项目集”）这一层才有一个小图标表示架构师。在敏捷转型的大潮中，他们可能一直心存疑惑，一方面，在敏捷培训中了解到，架构师这个职位大概是消失了，因为敏捷希望所有的架构问题都能通过整个团队成员来解决，通过团队的开发实践来演进；另一方面，普遍情况是，大型企业或大型项目或者项目集里仍然存在架构师这样的职位和职务，过去一直存在的组织结构在敏捷转型中也仍然不大会有太大的变化，因此架构师也仍然希望能够在敏捷转型中找到自己的位置，而且还能继续发挥过去的技术领袖及拥有审批权利的作用。业务分析师（Business Analyst，BA）的位置没有明确标注在图中，于是他们猜测可能会是 PO 或者是在项目集里的产品管理团队中，但是对在这样的组织结构下会承担什么样的职责感到疑惑。在很多公司进行的敏捷转型中，大多数情况下过去的项目经理都转变为 Scrum Master，因此项目经理在这张全景图中可以看到 Scrum Master 的位置因而比较确定和接受可能发生的职责变化。然而，我们没有看见项目集经理的图标，我们猜测大概就是位于全景图中间层中间位置的火车司机的图标——发布火车工程师（Release Train Engineer，RTE）。项目集经理通常是企业中一个相对高级的职位，需要管理多个项目的开发及发布。在全景图上却没有明显的人物汇报关系，这可能会让一些高级领导对推动这个模型有心理障碍，很多企业在自行试图采用这个模型的时候都有意无意忽略这个角色。在项目组合这一层上也没有出现一个项目组合经理的图标，代之以项目组合管理团队。反对这个模型的人会说以一个团队来代替一个人只是对敏捷团队职责的伪装，在这样的层级式管理下，不同层级中的人还是存在上下级的汇报关系，这是敏捷转型的潜在障碍。

SAFe 是一个讨论需求管理的敏捷方法论

还是回到这部著作书名中提到的敏捷软件需求。从本质上讲，我们推动敏捷转型的目的是让企业或者项目能够更快创造商业价值。因此，从识别业务价值到

通过软件开发、组织优化的开发团队及组织结构等一系列活动的根本目的是管理大型需求、加快发布有价值的软件、发布有质量保证的软件、减少企业或项目中的浪费、缩短价值流的发布周期。那么让我们再来审视一下这个规模化敏捷框架的全景图，把目光从全景图的左侧移到右侧：那里讨论的是对需求的层次化管理。从上到下是不同粒度的需求：篇章（Epic）→特性（Feature）→用户故事（User Story）。为什么对需求进行分层？因为在现实生活中存在很多大需求，这些需求最终变成很多大型软件系统。这部著作就是要讨论怎样用敏捷和精益的思想来开发“大需求”。我们事实上已经在使用很多大型软件系统，比如说 Microsoft Office（有人认为这是一个大集合，好，那我们就说 Microsoft Word）、Windows 操作系统、Mozilla Firefox（这只是个浏览器）、贝宝（PayPal）、腾讯 QQ（以及腾讯打造的相关生态系统，包括腾讯门户网站、微信、腾讯微博、在线游戏等）。这样的软件系统通常需要很多个 5～9 人团队共同工作才能开发出来。我们其实很容易发现，帮助一个 5～9 人团队实现敏捷转型还算是相对容易。我们已经有相对成熟的理论知识，也已经有很多成熟和资深的敏捷顾问和教练能够帮助团队发现问题和推动持续改进。大需求通常都有复杂的特性，这些特性可能来自多个产品团队，很多特性之间存在依赖关系，也存在优先级的冲突，这些特性甚至需求本身可能也存在冲突。这样的拥有共同商业价值的需求及特性会被放进一个共同的计划中，由多个开发团队紧密合作来实现。怎样才能让这个计划敏捷地实施，怎样才能让多个团队各自敏捷地工作，同时也能够让这个软件系统整体实现快速迭代式发布呢？大规模敏捷框架 SAFe 对这个挑战给出的解决办法就是提出三层需求管理模型，从项目组合开始，通过企业级的投资场景看板来提取出符合企业发展愿景的高优先级“篇章”，传递给项目集的产品管理团队来产生特性及通过优先级排序的特性待办事项表。项目集的产品管理团队是由项目集的产品管理人员和各个项目团队的产品负责人（Product Owner）组成，因此能够保证产品管理的一致性。同时本书还给出多种思考方法来帮助大型团队理解在何种情景下应用集权管理或分散管理，使项目团队能够有足够的授权来进行各自的开发工作，同时也能够对一些由单个团队很难处理的问题实行集中统一管理。

本书重点强调一个重要的敏捷思想，即按照节奏进行开发，按照业务需要进

试读结束：需要全本请在线购买： www.ertongbook.com

行交付（Develop on Cadence，Deliver on Demand）。各个项目团队具有一致的迭代周期和发布周期，通过在每个发布周期开始前全员进行集中敏捷发布规划（Agile Release Planning），使各个团队对马上即将开始的发布周期产生各自的开发计划，对齐相互的依赖开发或集成测试计划，找出可能存在的不一致之处，识别出可能存在的问题或风险并产生相应的风险减轻计划及应急计划。这个大规模敏捷框架希望能够通过实现这样的思想来帮助大型企业或大型项目来组织其更加敏捷的开发和发布活动。

本书可操作性很强。一边学一边用吧

简单地说，这部著作涵盖了 Dean 对大规模敏捷框架的完整思路及讨论，给出对企业或大型项目的敏捷转型和敏捷开发的需求管理解决方案。对软件开发企业领导层以及大型项目团队着手开展敏捷转型或者推动大规模软件敏捷开发有非常具体的管理策略及实施方法。在过去的几年里，我们也在帮助一些大型项目进行敏捷转型的辅导工作，SAFe 被这些大型项目或者企业采纳并且在实施过程中提高了他们的发布质量和工作效率。

我们非常乐于推广 SAFe，在 SAFe 不断取得成功的过程中享受应用 SAFe 带来的乐趣和个人及组织的知识积累与领导力提升。

译序 2

———博采众长，打造方法工具箱

（刘磊）

刘禹锡诗云：“千淘万漉虽辛苦，吹尽狂沙始到金。”又云：“流水淘沙不暂停，前波未灭后波生。”世界在不断变化之中。我们作为软件工作者体会尤深，技术日新月异，新的工程方法层出不穷。尽管如此，我们仍可以在软件行业地貌中发现一些热点，敏捷方式就属于这种热点之一。敏捷现在获得了广泛采用，虽然它已经走过十多年的历程，但是人们对它的探索与发展并没有终结。

敏捷提倡“拥有敏捷的品质，为行动做好准备，在行动中的机敏、活跃、灵巧”。已经有一些书介绍如何敏捷，尤其是如何组建敏捷团队，例如清华大学出版社的《Scrum 敏捷软件开发》、《用户故事与敏捷方法》、《硝烟中的 Scrum 与 XP》等。然而团队敏捷性无法自动促成企业级的敏捷，而且在大多数情况下团队只是作为一个开始，人们寻求建立精益、敏捷、可适应的软件企业，使得可以维持优越的竞争绩效。

本书介绍了一套精益敏捷的企业级软件需求实践，它适用于团队、项目集和企业层级。这套方法有来自软件工程论演变历程的根基，也是对一些重要敏捷方法的扩展（主要是综合 XP 和 Scrum），并且融合精益制造思想。本书适合作为大型软件企业实施敏捷转型的参考，也可以为中小企业或 IT 工作间提供敏捷实践指导。当然，还可以为初学者提供全面的敏捷知识，帮助他们了解方法论的演进。值得一提的是，我们可以借鉴 Leffingwell 所说的“敏捷软件物理学”的一些核心概念、底层起支撑作用的思想观念以及由相关的概念和结构形成的整个需求管理套路，根据具体环境打造定制的开发方法工具箱。

敏捷运动是软件工程发展道路上的又一高峰

敏捷运动的兴起源自业界对灵活可适应过程的需求，针对这种需求所进行的理论与实践探索可以上溯到上世纪 50 年代。在上世纪 90 年代中期的轻量级过程运动中，产生了现在很著名的一系列敏捷方法，包括 Scrum、极限编程和动态系统开发方法等。但是“敏捷”这一词汇是直到 2001 年发表敏捷宣言时才引入的，敏捷宣言也标志着敏捷初创阶段的完成。

敏捷不是时尚

敏捷方式经过十多年的发展，日趋丰富和完善。事实证明，敏捷不是时尚，根据美国 VersionOne 调查，18%的受访者从事敏捷项目超过 5 年，2～5 年的从业人员占总人数的 37%。75%的受访者反映生产力得以提高，管理客户优先级的能力和项目的透明度的提升尤为明显。

敏捷要求有企业文化的支持，根据 VersionOne 的调查，对于敏捷实施的最大障碍不是对方法学的认知，而是企业内部的文化。采取敏捷方式还必须使过程工具化，调查显示，当前最常用的敏捷项目管理工具包括：自动化验收工具、发布管理工具和持续集成工具。最常见的敏捷软件过程工具是标准的办公室效能工具，如 Excel；也有一些专门工具，如 VersionOne。缺陷追踪工具的采用也日趋增多。比较流行的开发工具包括 Excel、VersionOne、Microsoft Project、JIRA、Google Docs、HP Quality Center、IBM Rational Team Concert、Mingle 和 Verdor Y 等。

敏捷与传统方式有很大区别

敏捷是以一套敏捷实践取代传统方式中规定的各种里程碑评审和通过大前期设计阶段对需求的详尽说明。而且，敏捷是以适应性的过程取代传统方式中需求设计编码实现的僵化项目风格。在敏捷中的需求是一种过程，也就是贯穿开发周期地使用原型、实体模型、客户可以理解的语言等，通过与客户的对话协商来逐渐发现和充实需求。敏捷强调对过程的持续改进，即对 Sprint 评审与反思并进

行相应的调整。这是一项长期的任务，是用一些小的步骤来度量的一段旅程，这样的改进也没有简单的精确替代方案。这样的改进还包括：更好的编码实践、更好的单元测试和单元测试用例覆盖、功能测试自动化、持续集成以及其他改进的敏捷项目管理和软件工程实践。

当东方文化遇到西方思想

与敏捷同期发展的还有“精益软件运动”。精益制造是从丰田生产系统（TPS）发展而来，丰田生产方式诞生于二战后日本工业的艰苦年代，它有庞大的经济学和数学基础。丰田生产方式的哲学框架包含“一个目标”、“两大支柱”和“一大基础”。“一个目标”是低成本、高效率、高质量的生产，最大程度使客户满意。“两大支柱”是准时制和自动化。准时制（Just in time）生产是在必需的时候以必需的数量生产必需的产品。TPS 的自动化不是一般意义上的自动化，它是指人员与机械设备的有机配合，要求针对生产线上发生的质量、数量、品种的问题能够做到机械设备自动停机并有指示显示，发现故障问题的人员立即停止生产线，主动排除故障解决问题。“一大基础”是指改善，是要做到在追求目标的同时，通过改善得以实现量化管理、质量保证、对人尊重等效果。

精益软件开发运动是从精益制造发展而来，并且结合了在软件开发领域中一些精益的 IT 原则与实践，它是在敏捷社区中一种新兴的准精益的亚文化。精益软件开发这一词汇最早出现在 Mary Poppendieck 和 Tom Poppendieck 的同名书籍中。虽然精益方法的市场占有率目前还不高，但是精益方法具有雄厚的理论基础和丰富的制造业实践经验，根据本书作者 Leffingwell 的观点，精益软件开发似乎比敏捷更有前途。

精益敏捷实践可以提供有限制、生产式的能力

在本书中介绍的精益敏捷需求实践纳入了看板制度。如前所述，准时制是丰田生产方式的两大支柱之一，有些人认为 TPS 就是看板方式，这是错误的，TPS 是生产制造方法，而看板是实施准时制生产的手段，看板是一种生产排程制度。

看板在制造业的成功，启发了看板软件运动。对于软件开发来说，实施看板制度是一种管理方式的演进，它不是软件开发或项目管理的生命周期或过程。看板基于一个简单的思路，即限定工作任务，新任务的流入必须在现有任务已完成之后。看板意味着产生了可视信号，表明新的工作可以被拉取。这似乎是一个简单概念，但是看板带来的演变可能改变整个开发过程。

在实施看板制度的软件开发过程中，每项工作都要通过一系列的队列阶段，一项工作在一个阶段中完成之后将进入下游阶段。在下游阶段需要新的工作时，可以从上游拉取。在看板制度中仍然是试图快速交付新的价值，但是为此采取的方式却是限制每次承担的工作量。这样的限制有两种基本类型，即：质量限制和WIP 限制。WIP 限制是管理在任何时间处在任何状态中的工作数量，如果在某状态中的工作数量低于上限，则可以从上游“拉”取工作。如果工作数量到达上限，则必须等待处于此阶段的任何工作被完成并且被下游拉走。

看板排程可以提供明确的系统状态指示器。提高 WIP 意味着增加工作的未来交付时间，因为团队只能完成在生产力容量以内的工作。从管理方面来看，看板方式是对需求与供给的持续匹配活动，可以在正确的时间交付正确的工作。看板制度为一些更传统的团队、处于监管环境的团队、处于保守文化氛围的团队、工作在专业领域中的团队提供了通向敏捷性的途径。

精益敏捷实践可以帮助实现企业级的产品价值流

在本书中所介绍的一套精益敏捷的需求实践，正是对业界不懈推动的这两种软件运动的巧妙综合，它既借鉴了来自敏捷方式的实践经验，也运用并且是基于来自精益思想的一些原则。精益制造的核心思想是最大化客户价值，同时降低浪费。为此，精益思想改变了管理焦点，从优化单项技术、资产和垂直部门，改为优化生产与服务中贯穿技术、资产、部门到客户的水平流向的完整价值流。这种价值流是在一系列步骤之间存在的平稳连续的流动价值，实现企业级的产品开发流，可以避免在每次有新倡议时，启动和停止项目引入的延迟与开销。这意味着必须在过程组织中提供可见性和透明性、采取在制品限制、主动管理队列长度，并且开展为了建立和控制连续价值流所需的其他工作。

对浪费的消除则是沿完整价值流（而不是在一些孤立点）进行，产生需要更少人工努力、更少空间、更少成本以及更少时间的过程，使得与传统业务系统相比，产品与服务的成本更低，并且有更少的缺陷。从而使企业能够以更高的可变性、高质量、低成本和高效快速的生产周期，响应客户要求的变化，而且信息管理变得更简单和精确。

打造自己的方法工具箱

虽然您可能已经是某种方法的老手，或者对许多方法都有一定理解，再或者对于敏捷方式您还只是菜鸟，无论是哪种情况，通过对本书的深入学习，都可以为您带来对方法学变化历程和演进趋势的基础理解，有了这样的理解，可以帮助您面对由形形色色的现代软件方法形成的庞大生态系统。

敏捷方法与精益的对比

“精益”是一个伞形术语，它覆盖基于精益制造和丰田生产方式的任何工作方式，这包括精益建造、精益实验室以及精益软件运动等。

“敏捷”也是一个伞形术语，它覆盖一组软件开发方法学，包括 Scrum 和 XP 等，这些方法共享一些核心原则。

现在的许多敏捷方法的发展，受到了精益制造思想的深刻影响。在精益与敏捷之间存在许多共享，包括：以人为中心的方式、授权团队、适应性的计划、持续改进。有些敏捷专家强调缩短周期时间的重要性，也是受到精益的强烈影响。所以在软件世界中，精益与敏捷是深深交织在一起的，如果你从事敏捷开发，那么就是在实践精益，反之亦然。

取各种方法之长

所有软件开发方法都对应着一系列的实践，这些实践要符合某种次序以及相应的输入与输出。有些实践很少被软件过程采用，那么它可能面临终结。一种常见情况是对某种方法的过分倚重，并不断产生一些系统设计文档，导致重型的、

试读结束：需要全本请在线购买： www.ertongbook.com