软件工程实践丛书

PEARSON
Addison Wesley

敏捷软件开发工具
——精益开发方法

Lean
Software Development
An Agile Toolkit

- 根据具体情况定制敏捷实践
- 找到并根除软件开发中的浪费
- 为专业人士提供实用技术

（美）Mary Poppendieck
Tom Poppendieck 著

朱崇高 译

Pearson Education

清华大学出版社

敏捷软件开发工具
——精益开发方法

Lean
Software Development
An Agile Toolkit

书中给出了7个基本“精益”原则，并讲解了如何使其适用于软件开发领域。同时，书中还介绍了22种“思考工具”，帮助您定制适合具体环境的敏捷实践。精益原则能帮助您：

- 通过迭代趋向完美：将软件开发看成一个不断探索的过程。
- 管理不确定性：通过将变更嵌入系统“尽量推迟决策”。
- 压缩价值流：快速开发、反馈和改进。
- 在保持协作的基础上授权团队和个人。
- 增强软件的完整性：提高一致性、可用性、可维护性和适应性。
- 着眼整体：解决开发人员分散多处的问题。

简而言之，本书能帮助您重新关注价值、流程和人员，使您在质量、成本、速度和效益方面取得突破。

作者简介

Mary Poppendieck是敏捷联盟的常务理事。她在IT业有25年以上的从业经历，是一位经验丰富的业务运营和新产品开发领导。她曾构建了3M公司最初的及时精益生产系统。

Tom Poppendieck曾构建商务班机导航设备的并发开发支持系统。他在软件产品开发及COTS实施方面也有着丰富的经验。Tom目前致力于协助软件组织应用本书所描述的精益原则和工具，提高组织的软件开发能力。

ISBN 7-302-07867-X

9 787302 078678 >

定价：22.00元

PEARSON Addison Wesley

文稿编辑：李 强
封面设计：陈刘源
读者邮箱：Book@21bj.com
信息网站：http://www.epress.cn
http://www.34.cn
http://www.pearsoned.com

软件工程实践丛书

敏捷软件开发工具

——精益开发方法

（美） Mary Poppendieck
Tom Poppendieck 著

朱崇高 译

清华大学出版社

北京

内 容 简 介

精益思想已在制造、卫生保健和建筑等诸多行业取得了卓越的成效。敏捷软件开发更是让困境中的软件开发人员看到了曙光。本书揉合了两种思想的精髓，帮助读者将广为接受的精益原则转换为适应具体环境的敏捷实践，从而提高组织的软件开发能力。

本书为软件开发领域的开发经理、项目经理和技术主管编写，为其提供了大量的实用技术和思考方法。

EISBN: 0-321-15078-3

图书在版编目（CIP）数据

敏捷软件开发工具——精益开发方法/ （美）玻朋蒂克（Poppendieck , M.）， （美） 玻朋蒂克（Poppendieck , T.）著；朱崇高译. —北京：清华大学出版社，2003.12

（软件工程实践丛书）

书名原文：Lean Software Development: An Agile Toolkit

ISBN 7-302-07867-X

Ⅰ. 敏… Ⅱ. ①玻 … ②玻… ③朱… Ⅲ. 软件开发－方法 Ⅳ. TP311.52

中国版本图书馆 CIP 数据核字（2003）第 122106 号

出 版 者：清华大学出版社 **地 址**：北京清华大学学研大厦

http://www.tup.com.cn **邮 编**：100084

社 总 机：010-62770175 **客户服务**：010-62776969

文稿编辑：李 强

封面设计：陈刘源

印 刷 者：北京牛山世兴印刷厂

装 订 者：三河市李旗庄少明装订厂

发 行 者：新华书店总店北京发行所

开 本：185×260 **印张**：10.5 **字数**：246 千字

版 次：2004 年 2 月第 1 版 2004 年 2 月第 1 次印刷

书 号：ISBN 7-302-07867-X/TP・5714

印 数：1～4000

定 价：22.00 元

译 者 序

精益思想诞生于20世纪40年代末期。当时，由于缺乏足够的资金，刚成立不久的丰田公司制定了丰田生产系统，该系统的主旨是消除浪费。本书系统论述了针对软件开发的7个精益原则：消除浪费、增强学习、尽量推迟决策、尽快交付、授权团队、嵌入完整性和认识整体。在这7个原则中，消除浪费是最为重要的一个原则，它是其他原则的基础，也是它们的目的所在。

事实上，对任何企业来说，关键并不在于知晓消除浪费这一基本原则，而在于对浪费的界定，因为很显然浪费是任何企业都应该避免的行为，这是一个极为自然、浅显的道理。但在对浪费的认识和感知上，各企业之间存在很大的差异。在很大程度上，这种认识和感知的水平决定着企业的绩效。显然，丰田公司的成功意味着丰田生产系统及其对浪费的认识是值得研究和效仿的。但如果不考虑自身具体环境生搬硬套地移植其他企业的经验，注定将遭受挫折。对这一点，本书举出了不少事例，并进行了详细的讨论。事实上，这也是本书的价值所在：将精益原则与敏捷实践相结合，也就是说，将精益原则转换成适合于具体环境的敏捷实践，使空泛的原则成为具体的实施方法，让读者知道自己应该做些什么，采取哪些措施。

本书每一章对应一个精益原则，其核心内容是22个思考工具，这些工具旨在帮助读者将上述精益原则转换为适用于特定领域和环境的敏捷实践。此外，本书还穿插了一些相关案例和作者的亲身经历，读者可以方便自如地结合自己的实际情况领悟、应用和掌握上述7个精益原则，达到灵活运用的境界。

本书作者Mary Poppendieck和Tom Poppendieck均在IT领域从业多年，积累了丰富的经验，是精益软件开发方面的顶级专家。相信这本精心编写的著作将成为软件开发领域的一个里程碑，因为作者创造性地将精益原则与敏捷实践结合起来，试图开创一种全新的软件开发管理方法。

本书适用于软件开发领导者，同时也能供各类软件开发人员阅读参考，提高自己的业务能力。

为了提高我国软件业的管理水平和开发效率，清华大学出版社引进了这本名著，本人有幸组织翻译工作。参加本书翻译工作的人员还有朱崇贵、张梦兰，另外唐振宇和刘嫦娥在多方面给予大力协助，使本书得以顺利完成，在此一并表示感谢。

由于译者水平有限，书中一定还存在不妥之处，敬请读者批评指正，以便在下一版时修订。

朱崇高

2003年1月

序一

2001 年 2 月，当“敏捷”一词被用作为极限编程（Extreme Programming）、水晶方法（Crystal）、自适应软件开发（Adaptive Software Development）和 Scrum 等方法学的统称时，敏捷工业传统在当时的环境下已呈现出一派生机。Womack，Jones 和 Roos 合著的 *The Machine That Changed the World*，Smith 和 Reinertsen 合著的 *Developing Products in Half the Time* 以及 Womack 和 Jones 的 *Lean thinking* 已在我的书架上摆放多年。20 世纪 90 年代初，一些制造商成立了“敏捷论坛”。有关敏捷和精益工业产品开发的大量文献对我的著作 *Adaptive Software Development* 有着较大的影响。

不过，在本书中，Mary 和 Tom 将精益工业实践提升到一个新的水平——他们讲述了如何将这些实践直接应用于软件开发。研究制造工厂的价值流图与了解如何将这一想法应用于软件开发过程完全是两回事。同样，研究日本丰田公司基于集合的决策方法也与将这些方法应用于软件设计截然不同。Mary 在 3M 公司的工业产品开发经验使她能深入地了解这些实践的实际运作方式。而且，她与 Tom 在信息技术方面的阅历也使得他们深谙如何将这些实践应用于软件开发。

尽管敏捷软件开发的起源可追溯到 10 多年以前，但仅在两年前（以 2003 年初为准）它才形成一种潮流。将其与精益和敏捷工业产品开发相结合，可增强敏捷软件开发原理和实践的可靠性。而且，更为重要的是，它能提供强化敏捷实践的丰富想法。

举例来说，上述基于集合的决策与流行的设计决策方法是背道而驰的。传统工程（软件工程等）强调的是分析和早期决策，以便下游活动能井然有序地进行下去。基于集合的开发强调的是保持设计可选项的开放性，以便尽可能掌握更多的信息。这些信息不仅涉及某项特定的设计，而且还涉及所有设计的集成。基于集合的开发强调优化整体，而不是单项设计。简单设计和重构可对软件开发人员起到类似的作用——具备更多信息时，再制定某些设计决策。因此，基于集合的开发能提供一种并行性。这种并行性不仅能增强敏捷实践的可靠性，而且还能展示如何对这些实践进行扩展。

本书提供了大量将工业环境中的精益技术应用于软件开发的相关信息。尤其值得一提的是，它还为那些希望给自己的项目团队创造增值的项目经理、团队领导和技术经理介绍了一个工具箱，以免他们成为团队前进中的障碍。

Jim Highsmith
亚利桑那州弗拉格斯塔夫市
2002 年 3 月

序二

敏捷软件开发过程产生于20世纪90年代。它建立在经验、反复试验、对失败的认识以及最佳实践的基础之上。20世纪90年代初，我曾经在自己的软件公司采用过Scrum和极限之类的编程实践。在首次制定详细的Scrum实践时，我确保自己在每一种能想象到的开发情景中尝试它们。随后，我出版了自己第一本有关Scrum的著作。在缺乏Scrum和其他敏捷过程第一原理或理论框架的情况下，我试图确保它能真正发挥作用，以免蒙骗世人。

我和其他一些人曾试图为敏捷过程提供理论基础。我回头查阅了自己在工业过程控制理论方面的研究成果，我在杜邦先进研究所（DuPont's Advanced Research Facility）的朋友们曾帮助我理解和应用过这种理论。为了以类推法解释敏捷过程能发挥作用的原因，Jim Highsmith参照了复杂自适应系统原则和复杂性理论。

Mary和Tom Poppendieck为大家提供了一种更加易懂和稳健的日常框架，以便我们理解敏捷过程的运作机制。我曾经和他们一起出席过在意大利撒丁岛上举行的XP2002研讨会。当时，意大利特兰托大学经济学系主任Enrico Zaninotto作了题为*From X Programming to the X Organization*的主题报告。在该报告中，Enrico介绍了从简易车间过渡到装配线，再过渡到采用现代精益制造方法的制造迁移过程。他阐明了当前迫使人们采用精益制造方法的经济因素。报告结束后，Mary显然对这一论证感到非常满意。Enrico的报告使她将自己在制造业和产品开发方面的阅历与自己在精益建筑活动方面的所有合作以及对丰田生产系统的认识联系起来。

Poppendieck兄妹试图将所有这些活动和知识结合起来，本书就是这一努力的结果。在这项工作中，他们推出了一种构成敏捷过程基础的常识工具集。采用敏捷过程的读者可以参考Mary和Tom所描述的22种工具，以便了解最为常用的敏捷过程发挥作用的原因和方式，在对它们深入了解的基础上，对其进行修改，或构建属于自己的敏捷过程。本书介绍的这些工具即提供了这样一种框架。

能聆听Enrico的报告和分享Mary和Tom的思想结晶我感到由衷的高兴。长期以来，我们这一行业倍受责难，即我们应该能“像从事制造业一样从事这一行业”！这里提到的制造业是指弗雷德里克·泰勒和亨利·福特的产品装配线。我们按照泰勒原理构建的系统开发过程并不能有效运作，而且我们也不明白原因何在。Enrico笑着说：“现代制造业早在20年前就抛弃了泰勒原理！”

在对敏捷系统开发进行说明时，我们不再需要参考复杂自适应系统等深奥的理论和学科。我们可以查阅本书介绍的22种工具，并通过了解制造业和常识性知识掌握其基本原理。我们终于能利用某种专用工具为软件开发建模了！

Ken Schwaber

2003年2月

前　言

我曾经是一名非常不错的程序员。我编写的代码曾用来控制电话交换系统、高能物理研究、概念车的研发以及用来制造 3M 磁带的制造机和涂胶机。同时，我还擅长用 Fortran 和汇编语言编程，能迅速规定和构建微型计算机控制系统。

从事 10 多年的编程工作后，我随自己编写的某个系统进入了一家制造厂，并跻身 IT 管理人员之列。在这里，我掌握了材料控制、单位成本和生产数据库方面的知识。随后，quality-is-free[①]和及时生产潮流蔓延到这家工厂。这时，我才认识到几个简单的想法和得到授权的人员是如何改变一切的。

几年后，我涉足新产品开发领域，领导商业化团队开发嵌入式软件、成像系统以及光学系统。由于我热衷于新产品开发，我加盟了一家新兴的公司。后来，我创办了自己的公司，和产品开发团队，特别是那些从事软件开发的人员携手合作。

当时，我已经脱离软件开发行业 6 年之久。当我重操旧业时，对眼前的一切感到不知所措。面对项目管理学会（Project Management Institute，PMI）和能力成熟模型（Capability Maturity Model，CMM）认证计划，大家对最佳实践的理解似乎主要停留在过于强调过程定义和详细的前端计划上。更为糟糕的是，这些方法将我所熟知的精益制造潮流作为其存在的正当理由。

我强烈地意识到，精益制造的成功取决于深入理解什么能创造价值、为什么快速流程（rapid flow）是关键的，以及如何释放工作人员的脑力。在普遍强调过程和计划的现状中，我觉察到一种贬低这些关键性原则的倾向。例如，我听说，为了使“任何人都能编程”，需要进行详细的过程定义，而精益制造强调的则是培养一线人员的技能，并让他们定义各自的过程。

听说将事情“一次性处理妥当”的方法是花费大量时间并直接将需求放在第一位。对此我倍感惊讶。就我所知，当我试图控制一台机器时，使代码一次性成功运行的惟一方法是构建完整的模拟程序并无休止地对代码进行测试。我知道，交付给那家工厂的所有产品都经过了一整套测试，而“一次性处理妥当”意味着每个步骤都必须通过每一项测试。可以肯定地说，每个月市场都会需要新的装置或不同的磁带长度，因此那种在投产前固定产品配置的想法是闻所未闻的。这就是指定序列号的原因所在——这样我们就能辨别生产某一产品时的制造规格。我们从不指望本月生产的产品与上个月生产的产品完全相同。

详细的前端计划给我的印象是，它和精益制造原则完全背道而驰。在我看来，由员工

---

① Philip Crosby 在他 1979 年出版的 *Quality is Free* 中指出，质量是无价的，低劣的质量代价高昂。

组做出过程定义是与授权截然相悖，而授权是成功地进行精益制造的核心所在。我觉得，制造业的比喻似乎被滥用于软件开发领域。在我看来，CMM 在急于对过程进行标准化时忽略了发现和创新精神，而这种精神是成功地进行整体质量管理的关键因素。大家都知道，在制造业中，ISO9000 以及马尔科姆·波多里奇奖（Malcolm Baldrige Award）与质量计划的成功并无多大联系，或者毫无关系。它们对记录成功有用，但通常会妨碍创造成功。

对我来说，PMI 认证计划似乎在向新上任的项目经理传授若干针对软件项目管理的反模式，如工作分解、范围控制、变更控制、挣值、需求跟踪和时间跟踪等。在担任 3M 公司政府合同项目经理时，我对上述所有内容进行了了解，并强烈意识到它们给项目造成的浪费。当然，我们认为将它们应用到内部产品的开发中并非上上之策，在这类项目中，学习和创新才是成功的关键因素。

这并不是说 CMM 和 PMI 不好，只是对所有经历过精益革命的人来说，它们倾向于以错误的方式对待软件开发项目。我们希望本书能改变软件开发范式：将强调过程改为强调人员，将分散改为聚集，将推测改为基于数据的决策，将计划改为学习，将跟踪能力改为测试能力，将成本和进度控制改为商业价值的交付。

如果大家认为无法同时做到更好、更省、更快，那么，我们应该明白，当制造和产品开发尚处于前精益时期时，我们往往也会产生同样的想法。然而，我们认识到，通过强调价值、流程和人员，就可以提高质量，降低成本，加快交付速度。我们是在竞争对手夺走我们的市场时认识到这一点的。

祝愿大家在精益软件开发中处于行业领先地位。

Mary Poppendieck

致　谢

我们是本书的作者，但本书的思想则来自敏捷领域。数十年来，精益原则已在精益制造、物流和建筑领域取得成功。这些原则最终以敏捷软件开发的形式崭露头角，它们也是构成本书的框架。

本书的众多审阅者投入了大量时间精力深入阅读全书，并提供了有助于我们完善自己的思想和表达方式的反馈信息。他们是：Ken Schwaber，Jim Highsmith，Alistair Cockburn，Luke Hohmann，Martin Fowler，务实的 Dave Thomas，Bill Wake，Rob Purser，Mike Cohn 和 Martha Lindeman。感谢精益建筑学会的 Glenn Ballard 和 Greg Howell 对本书做出的贡献。我们还要感谢 Kent Beck，Tim Ocock，Ron Crocker 和 Bruce Ferguson 为本书付出的辛劳。

感谢我们的雇主、导师、团队成员、合作者和客户们。感谢所有参加我们的课程和辅导、提出探索性问题和向我们提供在各自领域行之有效（或无效）的精益原则实例的人们。最后，感谢我们所引用过的著作和文章的作者们，感谢他们为敏捷软件开发所做的贡献。

绪　论

本书是为软件开发领导者编写的一部思想工具书。它是一个工具箱，凭借这一工具箱，我们可以将得到广泛认可的精益原则转换为适合自己独特环境的有效敏捷实践。长期以来，精益思想在制造、医疗和建筑等各个领域取得了令人瞩目的成效。它是否能在软件开发中发挥相同的作用呢？显而易见的是，软件开发领域存在极大的改进空间。

Standish Group 的主席 Jim Johnson 向一位专心致志的听众[①]讲述了佛罗里达和明尼苏达两个州各自开发的州自动儿童福利信息系统（Statewide Automated Child Welfare Information System，SACWIS）的情况。在佛罗里达州，系统开发起始于 1990 年，当时预计将耗时 8 年，费用为 3200 万美元。据 Johnson 在 2002 年透露，佛罗里达州已花费 1.7 亿美元，估计系统将在 2005 年竣工，费用为 2.3 亿美元。与此同时，明尼苏达州于 1999 年着手开发基本相同的系统，并于 2000 年早些时候完工，费用为 110 万美元。开发效率的差距超过 200:1。Johnson 将明尼苏达州的成功归因于标准化的基础结构、最小化的需求以及一个由 8 名称职人员组成的团队。

不同的组织在从事基本相同的工作时会出现一些显著的绩效差距，而这只是其中一例而已。这类差距不仅出现在软件开发工作中，而且也会出现在其他诸多领域。公司之间的差距植根于组织历史和组织文化、对市场的态度以及它们抓住机会的能力。

人们对绩效高的公司与其一般竞争对手之间的差距进行了长期的研究，并对某些公司之所以比其他一些公司更成功的原因有了深入的认识。正如在软件开发中一样，根本不存在绝对灵验的方案。也就是说，没有银弹[②]。不过，我们拥有一些可靠的理论，它们能帮助我们判定哪些方法能产生高绩效，而哪些方法会阻碍高绩效的产生。制造、物流和新产品开发等领域已逐渐形成一种旨在为取得优异的绩效而营造最佳环境的知识体系。

我们注意到，有些早为其他学科废弃的方法仍被当作标准的软件开发实践。同时，视为标准方法的一些产品开发方法——如并行工程等——通常并未在软件开发中加以考虑。

也许，有些人之所以不愿采用产品开发方法，是因为过去采用了一些不恰当的比喻（metaphor）。软件开发业试图模仿制造业和土木工程的实践，结果必定会产生混乱。在一定程度上，这是因为他们对这些学科的真实性质缺乏充分的了解，而且他们并未认识到这

① 请参阅 Johnson 的 *ROI, It's Your Job*。
② 请参阅 Brooks 的 *No Silver Bullet*。

种比喻的限制。

虽然我们意识到滥用比喻的危害，但我们相信，软件开发类似于产品开发。而且，通过分析产品开发方法的改变如何使产品开发过程得到改进，软件开发界人士能掌握不少经验。从事客户软件开发的组织将认识到，他们的工作主要由开发活动组成。将软件作为产品或产品的一部分进行开发的公司应该从与之密切相关的产品开发中汲取经验教训。

佛罗里达州和明尼苏达州的 SACWIS 项目不禁使人想起通用汽车公司开发 GM-10 型轿车的事情来。该项目于 1982 年[①]展开。第一种车型别克 Regal 于 7 年后（即 1989 年）风靡街头，比预期晚了两年。GM-10 项目展开 4 年后，本田公司着手开发一种针对同一市场的新型雅阁轿车。这款轿车于 1989 年末上市，大约与 GM-10 Cutlass 和 Grand Prix 在同一时期面市。这两种轿车的质量情况如何呢？12 年后，基本无故障行驶 17.5 万英里后，我儿子仍然开着 1990 年产的雅阁轿车。

当时的一些研究[②]表明，在若干汽车公司中，与传统方法相比，日本汽车制造商所特有的产品开发方法使工程支出（engineering effort）降低了 50%，并将开发时间缩短了 1/3。这些结果与当时的传统观念是相抵触的，因为传统观念认为，在最后生产阶段进行更改所付出的代价要比设计阶段高出 1 000 倍[③]。人们普遍认为，快速开发意味着仓促决策，因此缩短开发周期将导致许多后期更改，从而增加开发成本。

为了防止更改费用呈指数增加。美国汽车制造商传统的产品开发过程都具有一定的顺序，而且与供应商的关系也限于正常交易关系。采用这一方法的结果是它会大大延长开发周期，同时无法在开发阶段后期做出适应当前市场趋势的调整。相反，本田和丰田等公司优先考虑的是快速的并行开发以及在开发周期的后期阶段进行更改的能力。为什么这些公司并没有因在后期开发阶段进行更改而付出巨大的代价呢？

为了避免在最后生产阶段因更改而遭受重大损失，一种方法是首先做出正确的设计决策，然后消除后期更改的需要。这是底特律的方法。丰田和本田公司摸索出另外一种方法来避免因错误的设计决策而造成的损失。首先不要做出无法逆转的决策。尽量推迟设计决策，在进行决策时，利用最为完善的现有信息做出正确的决策。这一想法类似于丰田开创的及时制造思想：在接到客户订单之前不要决定制造什么；不过，在接到订单后尽快生产。

推迟决策并非问题的全部，它只是以不同方式进行思考从而形成新的产品开发范式的一个例子。在 20 世纪 80 年代，通用公司和本田公司之间还存在其他差异。通用公司倾向于将关键性的决策推给几位高层权威，而本田公司新雅阁引擎的设计决策产生于对引擎罩和布局尺寸应该增加或减少多少毫米的工程级详细讨论。通用公司采用顺序过程进行产品开发，而本田公司采用的是并行过程，让那些负责制造、测试和维护汽车的人员参与汽车

① 请参阅 Womack，Jones 和 Roos 的著作 *The Machine That Changed the World*，第 110 页。
② 同上，第 111 页。
③ Thomas Group，*National Institute of Standards & Technology Institute for Defense Analyses*。

设计。通用公司的设计方案将接受营销经理和强势职能经理的修改，而本田公司则让一位领导者单独构思并不断将自己的构思传达给从事这项工作的工程师们。[①]

20 世纪 80 年代以本田公司和丰田公司为例的产品开发方法——通称精益开发——在 90 年代为众多汽车公司所采纳。如今，汽车制造商之间的产品开发绩效的差距已显著缩小。

精益开发原则已在汽车行业经受住了考验，而汽车行业设计环境的复杂性与大多数软件开发环境大致相同。此外，精益开发在很大程度上借鉴了精益制造理论。因此，精益原则大体上已为软件开发之外许多学科的管理者们所理解和验证。

精益原则、思想工具和敏捷实践

本书涉及的是精益原则在软件开发中的应用。人们对精益原则已经有了很多的了解，但值得注意的是，组织在应用它们时并不一定会取得成功，因为精益思想要求改变文化和组织习惯，这是某些公司无法做到的。另一方面，一些已经理解和采纳精益思想的公司实现了显著、持久的绩效改善。[②]

原则能指导对某一学科的思考和领悟，而实践则是为执行原则而采取的实际措施。[③]原则具有普遍性，但要了解它们如何适用于特定的环境并非总是那么容易。另一方面，实践则能对行动给予特定的指导，不过，需要针对某一领域对它们进行调整。我们认为“最佳”实践之类的事情是不存在的；实践必须考虑到具体环境。事实上，将其他学科的比喻应用于软件开发时产生的问题，通常是试图植其他学科的实践而非原则的结果。

软件开发是一门内容广泛的学科——小至网页设计，大至卫星入轨，都涉及到软件开发。某一领域的实践并不一定适用于另一领域。不过，原则能广泛适用于各种领域，只需将其转换为适用于各领域的实践。本书集中论述了将精益原则转换为敏捷实践的过程，并对这些实践进行了适当的剪裁以适用于特定的软件开发领域。

本书的核心内容是 22 种思考工具，这些思考工具能协助软件开发领导者开发出在其特定领域内最为有效的敏捷实践。本书并不是对敏捷惯例的详尽说明，它旨在帮助大家设计出适用于各自领域的敏捷实践。

要在组织内确立一种新的想法，必须具备两个先决条件：

- 必须证实该想法在实际操作时能发挥作用。

---

① 请参阅 Womack，Jones 和 Roos 的著作 *The Machine That Changed the World*，第 104～110 页。

② 例如，克莱斯勒采用了一种供应商管理的精益方法。在 20 世纪 90 年代，这一方法为该公司的振兴起到了重要作用。请参阅 Dyer 的 *Collaborative Advantage*。

③ 请参阅 Senge 的著作 *The Fifth Discipline*，第 373 页。

- 考虑采纳这一变更的人必须了解它起作用的原因所在。[①]

敏捷软件开发实践已被证实可以在某些组织内发挥作用，Jim Highsmith 在 *Adaptive Software Development* [②]一书中详述了这些实践之所以起作用的理论基础。本书通过将众所周知，并得到认可的精益原则应用于软件开发，进一步扩充了敏捷软件开发的理论基础。不过，本书还进行了更为深入的论述，它提供了一些思考工具，以便协助读者将精益原则转换为适合于特定领域的敏捷实践。我们希望本书能够促成对敏捷开发方法更为广泛的认可。[③]

内容指南

本书一共 7 章，它们分别阐述了 7 个精益原则和将每个原则转换为敏捷实践的思想工具。在结束本文之前，我们将对这 7 个精益原则进行简短介绍。

(1) **消除浪费**。浪费是指不能为产品增加价值的任何事物，该价值是指客户所能感知的价值。在精益思想中，浪费这一概念相当费解。假如某个组件被闲置，这就是一种浪费。假如在开发周期内收集的成册需求处于尘封状态，这就是浪费。假如制造商生产了超出当前需求的东西，这就是浪费。假如开发者为当前不需要的特征进行编码，这就是浪费。在制造过程中，随意搬运产品是一种浪费。在产品开发中，将开发工作从一个团队移交给另一个团队是一种浪费。理想状态是，摸清客户需要什么，然后生产或开发它，并尽快交付他们确切需要的东西。任何妨碍迅速满足客户需要的事物都是一种浪费。

(2) **增强学习**。开发是一个发现过程，而生产则试图减少变化。因此，精益开发方法会产生完全不同于精益生产的实践。开发就好比制定菜谱，而生产则类似于做菜。菜谱由经验丰富的厨师设计，这些厨师养成了探查什么能起作用的直觉以及使现有配料适合宴席需要的能力。不过，即便是技艺高超的厨师，当他们试图通过反复尝试，以确定能产生美味且易于重现这一美味的菜谱时，也会将这道新菜烹制出若干不同的口味。我们不能指望厨师们一开始就制定出完善的菜谱；作为学习过程的一个阶段，也应该允许他们针对某一名目烹制出若干不同的菜肴。[④]最好将软件开发设想成一种类似的学习过程，只是这一过程存在更大的挑战性，即开发团队规模大，其结果也比菜谱复杂得多。改善软件开发环境的最好方法是增强学习。

---

① 请参阅 Larpé 和 Van Wassenhove 的著作 *Learning Across Lines*。

② 请参阅 Highsmith 的著作，*Adaptive Software Development*。

③ 敏捷软件开发方法包括自适应软件开发（ASD，Highsmith，2000 年）、水晶方法（Cockburn，2002 年）、动态系统开发方法（DSDM，Stapleton，2003 年）、特征驱动开发（FDD，Palmer 和 Felsing，2002 年）、Scrum（Schwaber 和 Beedle，2001 年）和极限编程（XP，Beck，2000 年）。请参阅 Highsmith 的 *Agile Software Development Ecosystems* 查看敏捷方法的概括性介绍。

④ 请参阅 Ballard 的 *Positive vs. Negative Iteration in Design*。

(3) **尽量推迟决策**。推迟决策的开发实践在涉及不确定性的领域中行之有效，因为它们能提供基于选择的方法。面对不确定性，大多数经济市场都会制定出一些可选方案，以便为投资者提供一种避免在未来更加临近和易于预测之前锁定决策的方法。推迟决策之所以重要，是因为当决策建立在事实而非推测的基础之上时，就能做出更好的决策。在一个不断演变的市场中，保留可选设计方案要比早期做出承诺更有价值。在开发复杂系统时，推迟承诺的关键性策略就是将可更改性嵌入系统。

(4) **尽快交付**。直到最近，快速软件开发才得到人们的重视。在这之前，采用谨慎、完美的方法似乎更为重要。但如今，是时候将“速度优先”与“质量优先”列于同等的地位了。[①]快速开发具有很多优点。没有速度，就不可能推迟决策；没有速度，就没有可靠的反馈。在开发过程中，发现周期是学习的关键所在：设计、实施、反馈和提高。这些周期越短，能学到的东西就更多。速度能确保客户得到他们当前所需要的东西，而不是他们昨天需要的东西。速度还允许客户推迟决定他们真正需要的东西，直到他们了解了更多的情况。尽量压缩价值流是消除浪费的基本精益策略。

(5) **授权团队**。一流的执行过程取决于对细节的修正，没有人比实际承担工作的人员对细节更为了解。让开发者涉足技术决策细节对取得优异成绩至关重要。第一线人员能把对琐细问题的认识与许多人的才智结合起来。当他们具备必要的专业技能并得到领导者的指导时，他们就能做出更好的技术决策和过程决策。任何人都无法代替他们做出这样的决策。由于决策推迟和执行加快，中央集权无法协调组织工作人员的活动。因此，精益实践将采用拉动技术(pull technique)来制定工作进度，并包含本地消息传递机制(local signaling mechanism)，以便工作人员能相互了解需要做些什么。在精益软件开发过程中，拉动机制是一种协议，通过该协议可以定期交付日益精化的工作软件版本。通过可视图表、日常会议、频繁集成和综合测试达成本地信号的传递。

(6) **嵌入完整性**。当系统用户想道“没错！这就是我所需要的东西。他们猜透了我的心思！”此时就可以认为该系统具有完整性。市场份额可以粗略衡量产品的感知完整性，因为它能衡量客户在某一段时间的感知。[②]概念完整性表示系统的中心概念能作为一个稳定的内聚整体共同发挥作用，它是创造感知完整性的关键性因素。[③]软件需要具备额外的完整性水平——它必须在一段时间内维持其有用性。软件通常需要进行适当的演进，以便适应将来的需要。拥有完整性的软件具备一个内聚构架，具有很高的可用性和适用性，并能得到维护、调整和扩展。研究表明，完整性产生于明智的领导、相关专业知识、有效的通信和良好的训练。过程、规程和度量并不足以取代它们。

(7) **着眼整体**。复杂系统中的完整性要求熟知各领域的专业知识。在产品开发过程中，最为棘手的问题之一是，所有领域（如数据库或 GUI）的专家都倾向于尽量夸大产品某一

① 请参阅 Womack, Jones 和 Roos 的 *The Machine That Changed the World*，第 111 页。
② 请参阅 Clark 和 Fujimoto 的 *The Power of Product Integrity*，第 278 页。
③ 请参阅 Brooks 的 *Mythical Man Month*，第 225 页。

部分的性能（这一部分代表着他们各自的专业），而不是凝聚系统的整体性能。通常，如果人们优先考虑自己的专业兴趣，共同利益就会受到损害。如果以专业贡献而不是整体绩效对个人或组织进行衡量，就有可能产生局部优化效应。当两个组织存在合同关系时，这一现象尤其明显。因为人们理所当然会尽量夸大自己公司的绩效。在大型组织内，实施能避免局部优化效应的实践是一项艰巨的任务。而且，涉及合同问题时，这一任务的难度将大大增加。

本书为软件开发部经理、项目经理和技术领导而作。它围绕精益思想的 7 个原则组织自身内容。本书每章讨论一个精益原则，并提供一些思考工具，以便将精益原则转换为适合特定领域的敏捷软件开发实践。每章结尾针对在软件开发组织内实施精益原则提出了一些实用性建议。最后一章是使用该工具箱中思考工具的说明和保证书。

目　录