HZ BOOKS
PEARSON Addison Wesley

华章程序员书库

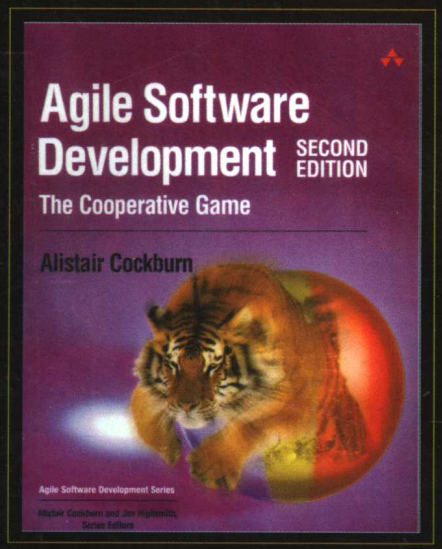
敏捷软件开发

（原书第2版）

Agile Software Development

The Cooperative Game

（Second Edition）



（美）Alistair Cockburn 著
苏敬凯 等译

“这是一本激动人心的书。作者以其丰富的实践经验，为我们提供了一部融合敏捷方法的力作。”

——Tom Gilb，著名的软件工程和系统工程专家

机械工业出版社
China Machine Press

程序员书库

敏捷软件开发

（原书第2版）

Agile Software Development

The Cooperative Game

（Second Edition）

（美）Alistair Cockburn 著
苏敬凯 等译

机械工业出版社
China Machine Press

本书揭示了敏捷软件开发的真正内涵。全书以“软件是创造和沟通的合作博弈”为中心向读者展示一个看待软件开发的崭新视角。全书共13章，包括创造和沟通的合作博弈、个人、团队的沟通与合作、方法集、敏捷与自适应、以及Crystal方法集等内容。

本书适合软件开发人员、管理人员、架构师等技术人员参考。

本书版权登记号：图字：01-2007-1841

图书在版编目（CIP）数据

敏捷软件开发（原书第2版）/（美）库克伯恩（Cockburn, A.）著；苏敬凯等译. —北京：机械工业出版社，2008.1

书名原文：Agile Software Development：The Cooperative Game，Second Edition

ISBN 978-7-111-23166-0

Ⅰ.敏…　Ⅱ.①库…　②苏…　Ⅲ.软件开发　Ⅳ.TP311.52

中国版本图书馆CIP数据核字（2007）第205693号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）

责任编辑：李南丰

北京牛山世兴印刷厂印刷·新华书店北京发行所发行

2008年1月第1版第1次印刷

186mm×240mm · 25.5印张

标准书号：ISBN 978-7-111-23166-0

定价：52.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

译 者 序

作为《敏捷软件开发宣言》和《相互依赖声明》的合著者，Alistair Cockburn的这本书揭示了敏捷的真正内涵。

“软件是创造和沟通的合作博弈”，这一隐喻向我们展示了一个看待软件开发的崭新视角，它告诉我们：软件开发的首要目标是交付有用的、可工作的软件，次要目标是为下一轮博弈做好准备。因为“编程就是构建理论”，而“理论”中的大部分意会知识不能或很难用文档来传达，而要靠沟通来传达，所以提高信息沟通的效率就提高软件开发效率的最有效的方式。

人不是生产线上的智能机器，人的特点是：人在沟通、学习和解决问题方面都不同于智能机器。以管理生产设备的方式管理人，显然不能最大限度地发挥人的潜能。

软件开发团队是一个交付软件的生态系统。团队的成功依赖于合作、沟通和协调。方法集就是你的团队同意遵循的那些协约。不同的协约适合不同类型的项目，没有放之四海皆适用的方法集，设计方法集时要“针对情形而制定策略”。

敏捷意味着有效率并且灵活机动。敏捷的过程既轻量又充足，轻量使它机动灵活，充足使它能够继续进行博弈。要不厌其烦地进行反思，才能使方法集自适应。

本书在论述敏捷软件开发的同时，还详细讲解了如何在团队内跨越沟通鸿沟交流信息，如何建设团队，方法集设计的常见错误和设计原则，敏捷的最佳击球点，在运行中构建和优化方法集以及反思研讨会等内容。本书还详细地讲解了XP和Crystal家族等敏捷方法集。

在书中，作者不仅把敏捷的思想推广到软件开发之外的领域，而且还深入地分析了对于敏捷的常见误解，全面回顾了各种敏捷方法集和理论的最新发展，并介绍了组织级敏捷方法集的设计，以及“最佳击球点和下降”、“敏捷与CMMI和ISO9001”等由来已久的问题。

本书的内容深厚宽广，论述每个问题时都提供了生动的例子或案例。本书的附录也不应被读者忽略，其中的《敏捷宣言》、《相互依赖声明》，还有宫本武藏的《五轮书》片段，都能很好地帮助读者理解敏捷、理解本书的内容。

感谢机械工业出版社华章公司的编辑一直以来对我们的翻译工作的支持和鼓励。能够翻译这样一本荣获Jolt大奖的书籍，是译者的荣幸。翻译本书的过程同时也是一个学习和提高的过程。

参加本书翻译的还有尚计升、史红伟、祝国志、张峰峰、张文军、张艳军、王小财和周宝华、张勇、杜兴平、刘志飞、沈海峰、谢扣林、乔义峰、刘查强、王义强、刘国际、杨传辉、王建华、汪明军、朱兆涛、毛付安。由于时间仓促，译者水平有限，翻译中的错误和不妥之处在所难免，欢迎广大读者批评指正。

译者于北京

2007年10月

第2版前言

2001年，软件开发的敏捷模型给世界带来了一场风暴。一年中，全世界出现了很多关于敏捷的著作和会议。在5年中，它影响了每一件事，从项目管理和公司经理们如何编写与客户的合同，到军事采购的过程，甚至到大学的课程。

是该看一下都发生了什么变化的时候了，看一下我们能从敏捷模型和（更具普遍意义的）合作博弈中学到什么。

在2001年2月写《敏捷软件开发的宣言》的时候，我已经深入到了编写一本关于“软件开发就是合作博弈”和“剪裁方法集以适合不同的项目”的书之中。这一宣言只不过是对我和其他人已经在做的事情的附和。

敏捷模型给世界带来了风暴。数以百计的开发人员在网上的签名板上签名（最初的非正式的敏捷联盟，不要与后来成立的敏捷联盟公司相混淆，后者负责管理在美国的召开的敏捷大会），以表示他们与敏捷宣言的价值观和原则一致。

在最初的关于“这是什么意思？”的问题之后，人们会问：

- “它什么地方适合所有开发的各种情形？”
- “我们如何将这些思想与其他思想混合在一起？”
- “我们如何将这些思想扩展到其他领域？”

在第2版中，新增的文字就是来解决这些问题的。

合作博弈模型在与敏捷模型一起成长。最初构建它是为了解释软件开发，但它引起了商业人士的共鸣，他们肯定地发现：大多数商业也是一个创造和沟通的合作博弈（并且是一个竞争博弈）。

你可以想象得到，在过去的这5年里，有些事情的出现使我感到惊讶。我着重说四点：

- **工程化**也是一个创造和沟通的合作博弈。经过一番重新表达之后，我们现在可以将软件工程定位为工程化领域中严格意义上的一员。

在第2版中，我只字未改地保留了我最初写下的那些反对“将软件开发看作是工程”的思想相当激烈的话。之所以这样做的原因是：对于大多数仍然错误地看待工程学的人来说，这些话仍然适用，它能把人们从相应的管理软件开发的错误方式中拉出来。在第2版的文字中，我把研究方向转回到“什么是工程化”的问题上。

在第二次世界大战之后，对于应用物理学的学科上的嫉妒造成了工程研究领域的一厢情愿的想法，对于工程学的流行的理解脱离了轨道。在重新研究了工程学实践之后，我们注意到：它本身就是一个创造和沟通的合作博弈。由此，我们可以创建一个更新版的软件工程的概念，它在实践上和教育学上都有意义。

- 在最近五年中，**用户体验设计**这一专业领域已经产生了强有力的扩展。过去，它被宣言的作者们忽略了，现在，它应该受到大家认真的关注。

所有这些早期的方法集都完全跳过了这一问题，可能是因为没有一个敏捷宣言的作者在这个领域有很强的专长。这个领域一直是并且还将继续是一个充满争议的领域，几乎没有可靠的结论。我将在这一节中讨论这些结论和争论的观点。

- 我已经学会了如何将**项目管理策略**与方法集区分开。

我们往往会把项目管理的策略当作是公司的过程或方法集的一部分，这是错误的。把那些应当是根据具体情况而制定的策略编写成一个固定的政策，这往往会迫使管理者们制定出没有效率的策略决策，并极大地增加组织的成本。因此，学会区分这两者非常重要。

- 更加令人惊讶的是：自从写作本书的第1版以来，围绕着本书（“敏捷软件开发”）的所有开发都发生了。

这意味着有很多需要讲述的话题，还有很多关于敏捷软件开发的含义和实践的误解需要澄清。

因此，关于敏捷软件开发的演进一章（第5A章“敏捷与自适应：演进”）比其他演进章要长很多。这是自然的，也是合适的。我希望那些已经掌握一定敏捷开发知识的读者能够在这一章中发现一些新的信息闪光点，而那些被其中一些问题所困惑的敏捷开发的新手能够得到一些清楚而深刻的认识。

阅读此书

阅读第2版的方式与第1版一样。

那些对“博弈规则”（包括但不限于软件开发）感兴趣的人，他们将发现前4章（包括引言）是他们的主菜。

那些想要迅速为自己的团队获得软件开发的敏捷模型和方法集构建的人，他们可能想直接跳到本书的第2部分，（或者附录A的敏捷宣言，然后从那里再）进入方法集的章节。

有些人读本书的目的是阅读Crystal方法集家族的基础知识。他们应当从第6章开始，我已经用改进过的这一方法集家族的描述对这章进行了更新。然后，他们可以返过头来再阅读关于沟通和合作博弈的部分，这是Crystal方法集家族的根本。

那些已经阅读了本书第1版的人，他们可能会想读每个“演进”章节。这些新的章节在页边上印有灰色的印签，很容易找到。我已经把本书的其他部分中的修改限制为仅仅是纠正错误，因此你不必为了寻找新的观点而重读每一页。

无论你如何阅读本书，我都建议你先读一下附录B，那里包含了宫本武藏（17世纪的武士）、Peter Naur和Pelle Ehn的几篇关键文章。

经过这5年，我变得更加坚定地感谢他们的这些文章了。现在，在敏捷培训的第一个小时中，我会讲述宫本武藏，使用Naur的理论作为描述软件开发的一个核心元素，定期重温

Ehn的文章以帮助我使自己的观点变得更加成熟。我猜你也会从它们身上发现类似的价值。

致谢

在修改文字方面，Silicon Valley Patterns Group的评审者们又一次非常周到地帮助了我。他们发现了错误和矛盾之处，在极少的情况下，我决定否定他们的意见，我把错误完全归咎于我自己。

特别是Russ Rufer和Tracy Bialik，他们几次坚持要求我确认我的意思就是我写的内容。感谢他们，我删除了两个连自己都不能支持的断言。他们还帮助我改进了几个图的解释。

其他的评审者友好地阅读了本书的草稿，并把希望、评论和修改发给了我，他们包括：John Rusk，Paul Old-field，Brandon Campbell，Romilly，Dave Churchville，Ilja Preuβ，Géry Derbier，Mark Levinson和Paul McMahon。John的agileKiwi.com网站是一个很出色的资源，上面有更多关于burn-up图和挣值图（还有其他敏捷话题）的信息；Dave的ExtremePlanner.com网站关注于敏捷的计划制定和跟踪软件；Paul关于敏捷和CMMi的看法很客观，因此我要求他贡献出来作为本书中的一段引用文字。

在很多场合下，为了使文章和页面的交叉引用按时完成，Celeste Rosell都会为自己的时间进度表而生气。我不知道她怎么得出的正确答案，使得段落中间不再会出现“???”。

Addison Wesley/Pearson Education的工作人员和他们的合同方对我都极有帮助。Carlisle出版社的Leslie Sweeney完成了一项杰出的工作：创建了一个清晰并且难忘的软件开发中的决策流的图（图1A-2）。Pearson公司的Alan Clements给我的《Crystal Clear》一书和本书的第2版带来了一个寓意深刻的封面设计。

高兴的事情是：盐湖城咖啡馆仍然供应土耳其咖啡，并且每晚都开到凌晨2点。在写作本书的第2版时，我依赖这两样东西。

不高兴的事是：当我无意中让crystalMethodologies.org网站过期时，一些在市场上钻空子的人占用了这个域名。因此，这个网站不再会提供有用的材料，当然也不再与我的工作有关系。我创建了Alistair.Cockburn.us这个新网站，它现在保存着我的所有文章、谈话和其他材料。幸运的是，我能够保持这个新网站完好无损并且有用。

第1版前言

软件开发是一门艺术、一门技艺、一门科学、一种工程，或者根本就是别的什么东西？这又有什么关系吗？

是的，它有关系，它与你有关系。你的行动及行动的结果会因为这些答案中哪个更正确些而有所不同。

主要的事情是这样：你希望尽快地完成你的软件，并且没有缺陷，但比这更重要的是：你需要有一种方法来检查你的团队是如何按照这种方法做事的。

目的

是该重新检查一下软件开发的基础概念的时候了。

麻烦的是：当查看项目时，我们所注意到的事情受到了自己知道应该注意的事情的约束。在流过我们的极其丰富的经验流中，我们学会了区分独特的和可分开的东西，并把这些东西从流中抽出来检查。在某种程度上，我们缺乏不同的关键辨别特征，同时也忽略了一些就在我们面前的事情。

我们使用一些词汇将这些辨别特征锚定在了我们的记忆中，然后用这些词汇来反思我们的经验。如果没有这些用来锚定辨别特征的词汇，我们就不能将自己的记忆投入到我们的交谈当中，也不能为应付未来而构建有意义的策略。

换句话说，为了能重新检查软件开发的基础概念，我们必须重新考虑那些用来划分我们的经验的辨别特征和那些用来锚定我们的记忆的词汇。

当然，对于任何一部书来说，这都是一个离谱的要求。这意味着本书的前面几章会相当抽象。但我也没有找到一个方法能够不这样做。

人们上一次为软件开发构建词汇表已是20世纪60年代后期的事了。那时，他们创造了“软件工程”一词。这个词既是一个期望，也是一个为未来指出的方向。

值得注意的是，在做出了“编程应当是一门工程”的论断的同时，Gerald Weinberg写了一本叫做《The Psychology of Computer Programming》的书。在该书中，Weinberg认为：软件开发根本不是一门非常像工程学的学科。它表现得更像某种非常以人为中心和以沟通为中心的东西。对于这两点，Weinberg的观察与人们在随后30年中所报告的结论是一致的，软件工程仍然只是一个一厢情愿的词汇。

在本书中，我将

- 建立一些用来谈论软件开发的辨别特征和词汇。

- 对于那些已经过于经常地被推到悬崖边上的软件项目，使用这些词汇来检查和确定它们的关键性方面。
- 把方法学的思想和原则作为“行为规则”来讲解。
- 将我们对于这些行为规则的需要与“每个项目都是独一无二的”的思想融合起来，从而推导出一些有实效并且自演进的规则。

我希望，在你读完这本书之后，能够使用这些新词汇来检查自己的项目，能够注意到一些你之前没有注意到的事物，并且能够把这些观察结果表达出来。熟练了之后，你就能够：

- 讨论极限编程、能力成熟度模型、个人软件过程或其他你喜欢的过程。
- 确定每个过程在什么时候较适用或较不适用。
- 理解那些观点不同、能力不同或经验不同的人。

读者

每位阅读本书的读者一定都有着不同的经验水平、阅读风格和角色。下面介绍一下：应如何根据经验、根据阅读风格或根据角色来阅读本书，才能使你获得最大的收益。

根据经验

本书是写给那些较有经验的读者的。书中没有包含那些在软件开发时应遵循的规程；实际上，本书的核心是“每项技术都有局限性”的概念。因此，不可能为软件开发指派一个最佳的正确方式。希望本书能够帮助你理解这点，从而为你引入一些关于“如何处理现实世界中的情形”的建设性观念。

如果你是中级的软件从业者，有一些软件项目的开发经验，并且你现在正在寻找你所了解的那些规则的边界，那么你会发现下面这些主题是最有帮助的：

- 哪种类型的方法集适合哪种类型的项目。
- 用于为项目选择适当的方法集分类的索引。
- 敏捷方法学背后的原则。

作为一个中级软件从业者，你应该认识到：在应用这些思想时，你必须加入自己的判断。

如果你是一位高级软件从业者，那么你已经知道了在适应性方面的所有建议。你可能正在搜索一些词汇来帮你表达它。你会在下面这些主题中发现这些词汇：

- 管理沟通的不完全性。
- 持续的方法集再创造。
- 敏捷软件开发的宣言。

即便是对于那些高级的软件开发者，也有一些新主题：用于描述方法集的词汇表和用

于即时方法学调优的技术。

根据阅读风格

与后面几章相比，前面几章比较抽象。

如果你喜欢抽象的内容，那么可以从头到尾地来读本书，观看这些抽象话题的演出，从而明白如何按照本书的思路来解决那些不可能的问题。

如果你想要尽可能快地得到一些具体的内容，则在第一次阅读时，你可以跳过前面几章，从第4章开始。然后再返过头来读那些关于“合作博弈”和“信息的对流”的章节，来获得这些新词汇的关键部分。之后再研究引言及关于个人和团队的那几章，以弥补前面两部分之间的脱节。

根据角色

对于软件开发的出资人来说，他们可以通过本书了解到：不同的组织结构、行为结构和资金结构如何影响他们从自己的开发团队收到价值的速度。项目出资人可能不像直接参与到项目中的那些人那么关注方法集构建的细节。但他们仍然应当理解某些方法论决策的后果。

对于团队领导和项目经理，他们能够看到：安排座位、团队组成和个人的个性如何影响他们的项目的产出。他们还能够学到：哪种干预更可能导致更好或更糟的后果。他们需要理解他们的方法集的建立和后果，以及如何改进他们的方法集——使它尽可能的轻，却又仍然足够。

对于过程和方法集的设计师，他们可以检验和争论我在用于方法集设计的词汇和原则上做出的选择。由此引发的讨论必将对这一领域大有裨益。

对于软件开发者，由于他们从事的是这个职业，所以他们会逐步理解全部这些内容。在从新手到领导者的一般过程中，他们将不得不注意到：哪些内容会对于他们的项目起作用，哪些不会。他们还不得不学会如何调整自己的环境以使其更有效率。“我们的方法集”真正的意思是“我们在当前项目的情况下要遵循的那些协约”，因此，“理解方法集构建的基础知识”已经成为了从事这个职业的每个人的责任。

本书的组织

在开始时，本书设定了两个几乎不可能的问题，而在本书结束时，则得出了这两个问题的答案：

- 如果从根本上说，沟通是不可能的，那么项目组成员该如何设法进行沟通呢？
- 如果所有的人和所有的项目都有所不同，那么我们如何为有生产力的项目创建一些规则？

为了实现这种设计，我在撰写本书的时候采用了一些解密式推理小说的写作风格。我从“什么是沟通”和“什么是软件开发”这两个最广义的最具哲学性的讨论开始。

在这两个讨论之后，仍然要讨论一些相当抽象的话题，比如“人类的性格特征是什么？”以及“什么影响了团队中的思想的变化？”

最终，本书进入到了较具体的领域，“方法集的基本元素和原则是什么？”。如果你希望早点读一些具体的内容，那么最好从这里开始。

最后，讨论进入到那些最具体的问题：“轻量级的、足够的、自演进的方法集是怎样的？”以及“一个小组如何能够即时地创建一个定制的敏捷方法集，从而对项目有所帮助？”。

在两个附录中，包含了一些支持材料。附录A包含了由17名经验非常丰富的软件开发人员和方法论研究者共同签署的《敏捷软件开发宣言》。

附录B包含了三篇文章的摘录，这三篇文章没有像它们应当的那样被广泛阅读。收录它们是因为它们是本书所描述的那些话题的核心。

本书思想的传承

本书中的这些思想所基于的是25年的开发经验和10年的对项目的直接调查。

1991年，IBM咨询集团请我去设计它们的第一个面向对象的方法集。我看了当时那些相互冲突的“方法学”的书籍之后，没有得到一点帮助。我的老板Kathy Ulisse和我决定：为了更好地了解他们真正工作的情况，我们应当听取项目团队的意见。令人大开眼界的是：他们所使用的词汇与那些书中的词汇几乎完全不搭界。

访谈一直那么有价值，我现在仍然会带着那些充分的、有趣的成功故事来探访项目，从而发现他们遇到了什么、学会了什么以及建议了什么。在访谈之前，我询问的至关重要的问题是：“你还愿意再次以同样的方式工作吗？”。当人们在描述自己的体验时，如果所使用的词汇与我的词汇表不匹配，那就表明这是一个我缺少辨别特征和词汇的新领域。

现在撰写本书的原因在于：这些词汇和辨别特征最终关系到对于项目实际情况和项目结果的描述。在诊断和干预方面，它们已经证明了自己比我之前使用过的任何工具都更有价值。

本书中的思想来自于几十个开发团队、八个方法集设计和我所参与过的很多成功的项目。

敏捷

我不是唯一使用这些思想的人：

- 20世纪80年代后期，Kent Beck和Ward Cunningham一直在研究那些在20世纪90年代后期被称为极限编程（XP）的技术。

- 在20世纪90年代中期，Jim Highsmith研究了在复杂的自适应系统中语言和业务的使用，并在他的《Adaptive Software Development》一书中写到了如何将语言应用到软件开发中去。
- 与此同时，Ken Schwaber和Jeff Sutherland正在构建Scrum开发方法，在同一时期，很多项目领导者都进行了类似的尝试来描述类似的想法。

2001年2月，我们这群人聚在一起讨论我们的差异和相似性。我们发现我们之间共同的东西多得令人惊讶。我们使用敏捷一词来描述我们的意图，并且写下了《敏捷软件开发宣言》（参见附录A）。

我们仍然在不断公式化我们所共享的原则，并且不断发现有很多其他人应该来参加这个会议，只是他们不知道这个会议或者他们的日程安排使他们不能出席这次会议。

敏捷软件开发的核心是：使用项目行为的“轻量但足够”的规则以及使用以人为本的规则及面向沟通的规则。

敏捷意味着机动性，这一特征在现在比任何时候都更重要。把软件部署到Web上使软件竞争比以往更加激烈。想要在行业中立足，不仅要交付软件并且降低缺陷，还要跟踪不断变化的用户需求和市场需求。不断在行业中取胜要求在软件开发这一博弈中取胜。而在博弈中取胜则依赖于对所要进行的博弈的理解。

我发现的对于“业务中的敏捷”的最好的描述来自于Goldman（1997）：

“敏捷是动态的、与情景相关的、有进取心地拥抱变更，并且是面向成长的。它讨论的不是：为了安然度过那些吓人的竞争‘风暴’，如何改进效率、压缩成本或关上业务舱口。它讨论的是成功和取胜；是新兴的竞争性场所中取胜；是在很多公司现在害怕的竞争风暴的正中心赢得利润、市场份额和客户。”

敏捷软件开发系列丛书

在近10年来关心软件开发中的敏捷的人中，Jim Highsmith和我发现了很多共同点，我们一起努力以一些相对轻量、有效率、人力的软件开发技术为基础，出版了一个敏捷软件开发系列图书。

我们的这个系列的图书以这两个核心思想为基础：

- 不同的项目需要不同的过程或方法集。
- 与关注过程相比，关注技能、沟通和集体能使项目变得更有效率、更敏捷。

这个系列有3个主要议题：

- 能够改进那些正在做某种特殊工作的人的效率的技术。这可能是一个正在设计用户接口的人、手机信息的人、制定项目计划的人、设计的人或者测试的人。无论是谁，正在进行这样一个工作就会想知道这个领域中最优秀的人如何完成他们的工作的。《编写有效用例》（Cockburn 2001c）和《GUIs with Glue》（Hohmann，即将出版）

就是两本个人技术的书籍。

- 改进一组人的效率的技术。这些可能包括团队构建、项目回顾、决策制定等等技术。《Improving Software Organizations》（Mathiassen 2002）和《Surviving Object-Oriented Projects》（Cockburn 1998）就是两本团队技术的书籍。
- 特别的例子、成功的敏捷方法集。那些想选择一个基础方法集来剪裁的人，会想要找到一种已经在类似的情形中成功使用过的方法集。修改一个现有的方法集比创建一个新方法集更简单，比使用一种为不同的情形而设计的方法集更有效率。我们盼望着能找到其他一些例子并将其出版。

这两本书是敏捷软件开发系列的基础：

- 本书，它使用我最喜欢的词汇表达了关于敏捷软件开发的思想：软件开发就是一个合作博弈，方法集就是关于协调的协约，还有方法集的家族。
- 第二本书是Highsmith即将出版的《敏捷开发的生态系统》。它扩展了关于软件开发中的问题的讨论，在《敏捷软件开发宣言》上签字的人们的不同建议中的共同原则，还有一些共同的敏捷实践。Highsmith的前一本书《自适应软件开发》，使用他最喜欢的词汇表达了他关于软件开发的思想，即一个复杂的自适应系统。

你可以在网络上找到更多关于Crystal、自适应和其他敏捷方法集的内容。具体的网站和话题都包括在本书最后的参考文献中。你可以从下面这些网站开始：

- www.CrystalMethodologies.org
- www.AdaptiveSD.com
- www.AgileAlliance.org
- 我的网站：members.aol.com/acockburn（http://alistair.cockburn.us）

致谢

Ralph Hodgson有一个很了不起的图书馆，里面有很多费解但有趣的图书。更令人震惊的是，他想办法放进自己的公文包的正好是我碰巧需要接着读的那些费解的书：Vinoed的《Sketches of Thought》和Wenger和Lave所著的《Situated Learning》。你在参考文献一节所看到的那些有趣但费解的书可能都来自Ralph的图书馆。

Luke Hohmann教给我了Karl Weick和Elliot Soloway。Jim Highsmith告诉我："突现行为（emergent behavior）"是规则的一个特征，而不只是"侥幸"。他们两个人都花了大量的时间来影响话题的顺序和参考的准确性，并且在每一页上都进行了评论。

Jason Yip很优美地把我所描述的信息传播比喻成气体弥散。他写道："Kim正在传送信息。信息就是绿色气体。Kim正在传送绿色气体……"。真可怕！你可以猜到我修改了这些句子！

Bo Leuf带来了一个很棒的双关语"argh-分钟"（替代"尔格-秒"），用它作为那些受挫

的沟通会话的度量单位。他也非常友好地反复核对了我的一些断言。例如，他写信给一些以色列人，核实我对于以色列人的观点："交谈中的礼貌会被认为是无礼，而不是恭维"。这引起了一连串令人兴奋的邮件交流，其中包括（一个以色列人写到）："作者对此的看法无疑是错误的……。在几天没见之后，我们总要彼此问候并握手……我认为作者把不能容忍工作上的错误误解成了缺乏礼貌"。另一个人说："考虑到你的热情。无论你说什么，都无可厚非。在我看来，以色列人要你有自己的看法并坚持你自己的看法。当然，他们也有自己的看法。至少有一个，:-)"。Benny Sadeh提供了我最终使用的词"坦诚"。

Martin Fowler为方法集的讨论贡献了"可视性"这一便于使用的概念。此外，他还提出了一些建设性的意见，并且用非常和蔼的态度指出了他认为很糟的地方。

我还要承认并且感谢其他一些积极的评审者：（按照字母顺序）Alan Harriman，Allen Galleman，Andrea Branca，Andy Sen，Bill Caputo，Charles Herbaut，Charlie Toland，Chris Lopez，Debbie Utley，Glenn Vanderburg，James Hanrahan，Jeff Miller，Jeff Patton，Jesper Kornerup，Jim Sawyer，John Brewer，John Cook，Keith Damon，Laurence Archer，Michael Van Hilst，Nick Fortescue，Patrick Manion，Phil Goodwin，Richard Pfeiffer，Ron Holiday，Scott Jackson，Ted Young，Tom DeMarco和Tracy Bialik。

我要加倍感谢Silicon Valley Patterns Group，他们这个小组不厌其烦地对草稿进行了仔细的分析。

感谢盐湖城产品团队的Elizabeth Wilcox，Cathy Gilmore，John Roberts和Malia Howland。他们在短得不合情理的时间内完成了把手稿变成最后的书籍这一难以置信的工作。

所有这些人都尽他们所能帮助我修改不好的部分并保留好的部分。如果我再有几年的时间来一直修订这本书，那么我甚至能够做到让他们全部都认可。

因为没有额外的这几年，所以我要感谢他们的努力，并且为没有改正所有的不当之处而向他们道歉。

感谢上帝，Beans & Brews咖啡店最终又开始播放爵士乐和摇滚乐了。我写作的这几个月都在听重金属和乡村音乐。感谢盐湖城的烤肉店，他们直到深夜还开着门。

为了避免我们将来发生尴尬，我的姓的读音是"Cō-burn"，是个长o音。

附加版权信息

出版公司（300 Massachusetts Avenue, Boston, www.shambhala.com）。

《守－破－离》作者Ron Fox，摘自《The Iaido Newsletter》第7卷第2册，54页，1995年2月。授权转载：Ron Fox, MSU Kendo Club for Shu Ha Ri。

对于eBucks.com和关于Crystal Orange Web的描述的引用。授权转载：Michael Jordaan, eBucks.com。

对于在开放源码的项目中的沟通模式的讨论的引用。授权转载：Mike Nygard。

图3-10授权许可：Joshua Kerievsky, Industrial Logic, Inc., www.industriallogic.com, Copyright © 2001.

图3-11授权许可：Ron Jeffries, Ann Anderson, & Chet Hendrickson, *Extreme Programming Installed*, Addison-Wesley, 2001.

图3-1、图3-9、图3-15和图3-16授权许可：Evant Solutions Corporation, www.evantsolutions.com, Copyright © 2001.

图3-2、图3-3、图3-6、图3-7和图3-8授权许可：ThoughtWorks, Inc., www.thoughtworks.com, Copyright © 2001.

图3-12和图3-13授权许可：Ken Auer, RoleModel Software, Inc., www.rolemodelsoft.com, Copyright © 2001.

目　录