软 件 工 程 技 术 丛 书

HZ BOOKS

PEARSON
Addison Wesley

前沿论题系列

敏捷软件开发生态系统

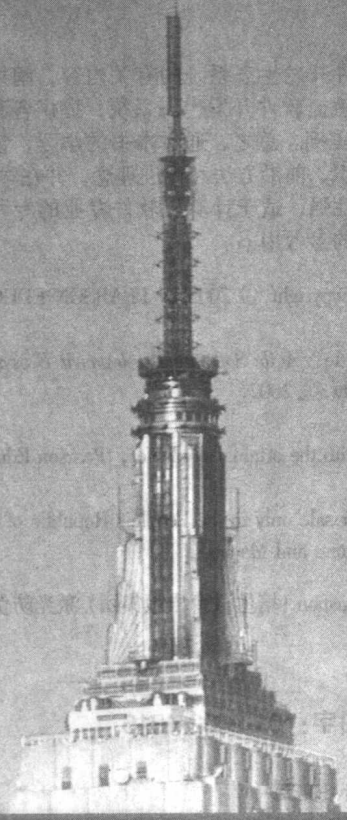
Agile Software Development Ecosystems

（美） Jim Highsmith 著
姚旺生 杨鹏 等译
徐锡山 审校

机械工业出版社
China Machine Press

软 件 工 程 技 术 丛 书

前沿论题系列



敏捷软件开发生态系统

Agile Software Development Ecosystems

（美） Jim Highsmith 著

姚旺生 杨鹏 等译

徐锡山 审校

机械工业出版社
China Machine Press

本书全面论述了敏捷软件开发生态系统的有关内容，阐述了变化驱动的信息时代经济的关键特征，介绍各个敏捷软件开发生态系统，提供各种类型的项目，展示了一个敏捷软件开发生态系统的实例。总之，通过本书的学习，读者能够了解到敏捷软件开发生态系统方面的基础知识、前沿方法和先进理念，并在学习和工作中受益。

本书简明、易懂、实用性强，适于计算机软件专业的专科生、本科生和研究生使用，也可作为相关从业人员的参考用书。

本书版权登记号：图字：01-2002-4816

图书在版编目（CIP）数据

敏捷软件开发生态系统/（美）海史密斯（Highsmith, J.）著；姚旺生等译．-北京：机械工业出版社，2004.1
（软件工程技术丛书　前沿论题系列）
书名原文：Agile Software Development Ecosystems
ISBN 7-111-12597-5

Ⅰ.敏…　Ⅱ.①海…②姚…　Ⅲ.软件开发　Ⅳ.TP311.52

中国版本图书馆 CIP 数据核字（2003）第 062105 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：杨　文
北京昌平奔腾印刷厂印刷·新华书店北京发行所发行
2004 年 1 月第 1 版第 1 次印刷
787mm×1092mm 1/16·20.25 印张
印数：0 001 - 4 000 册
定价：38.00 元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线电话：（010）68326294

译者序

随着科学技术的高速发展和因特网的普及应用，人们生活和工作的方式发生了很大变化。而随着我国加入WTO，竞争环境则越来越严酷，尤其是在IT行业。一个IT企业要想在这种环境下生存并发展，就必须采用新的管理方法和工作模式。

只要编写过软件，就一定会熟悉结构化、模块化，知道什么是自顶向下。CMM（能力成熟度模型）是一个象征公司软件开发能力的标志，ISO则是一个象征公司管理规范化程度的标志。在CMM和ISO体系中，为了达到最终的完美结果，需要预先制定出严格的工作步骤，然后再严格按步骤实施。预先制定严格的方法和步骤在很多企业十分有效，但在IT行业却可能成为例外。因为许多情况下要在项目的开始就确定用户的最终需求几乎是不可能的事情。一个成功的项目小组总是在和用户不断地交流，根据用户的建议不断调整自己的工作计划和目标，同时也以务实的态度来满足用户的需求。

因此，在软件产品的开发过程中，应该随时调整计划、方法和目标。或者说，为了交付使用户满意的软件产品，项目小组和开发人员必须具有“敏捷”特性。

敏捷软件开发方法以人为本，以满足用户需求为目标，其特征是：具有随时响应外界变化的能力。这种变化体现在许多方面：如经济全球化导致的业务和经济环境更难确定，竞争对手的策略和用户需求的不断变化，技术的高速发展，员工薪酬的不断提高，等等。采用敏捷方法的公司具有驾驭这种不确定性的能力，从而能立于不败之地，蓬勃发展。

敏捷软件开发生态系统包含了许多方法，这些方法构成了敏捷软件开发的总体。没有哪一种方法是最有效的，必须根据实际情况灵活运用。这种灵活性就是敏捷方法的精髓。

因此，如果你在开发软件并期望取得成功，就需要采用敏捷方法；如果你希望能在不确定的世界中把握自己的方向，就应该采用敏捷方法；如果你希望能压制住竞争对手取得更大的市场份额和商业利润，就必须采用敏捷方法；如果你想塑造一个团结协作、奋发向上、灵活高效的企业文化，最好采用敏捷方法；如果你想让用户对开发出来的工作软件十分满意，惟一的选择就是运用敏捷方法。

本书序、前言、引言和第一部分由姚旺生翻译，第二部分由杨鹏翻译，第三部分由张晓东翻译，其余部分由周飞和姚旺生共同翻译，赵德忠也参与了翻译工作。

国防科技大学计算机学院徐锡山教授审阅并修订了本书的译稿。贲可荣教授为本书的翻译出版做了大量协调工作，提出了许多建设性意见，在此一同表示感谢。

本书适合计算机行业的各类人员阅读。对于从事软件项目开发的管理人员和技术人员，本

书也是一本有助于开扩思路的好书。对于学习软件工程专业的本科生和研究生，本书可作为软件工程方法学教材或教学参考用书。

由于译者水平有限，且时间比较仓促，不当之处在所难免，恳请广大读者批评指正！

译　者

2002 年 12 月

序

20世纪90年代是IT业快速发展的10年。我们遵从着CMM和ISO体系，不仅仅追求软件构造方式的完美性，也注重总体的可预测性。我们的结论是，仅将事情做对是不够的。我们必须预定一个基调来预先准确说明打算做什么，然后再不多不少地严格执行。我们打算（按CMM的术语）“将工作计划好，并为完成计划而工作”。

20世纪90年代我们的工作重点在很大程度上和我们着迷于日本人的工作方式有关。回想1990年，回忆那时的研究主题，充满着一种失落感。西方人丧失了他们的机遇，日本人则能把握将来。毕竟他们工作更努力，经常加班加点，有极为严格的纪律，并有严格的质量等级制度，这对其他人来说，只能是一种梦想。日本现象，所谓“太平洋虎”，就是那时发生的一切。我们当时认为，如果要生存，最好像日本人那样工作，所以我们采取了更加日本化的方式：严格和过程化。

但在20世纪90年代，日本的发展并不一帆风顺。在90年代的最后几年日本经济急剧下跌，导致了大衰退。这说明，努力工作、守纪律、高效率、严格，这些80年代适用的内容并不是90年代的取胜法宝。20世纪90年代所要把握的就是——灵活。因特网改变了一切，只有那些随时准备应付这些快速变化的人，才能赢得胜利。敏捷是关键所在，其余的并不重要。

类似地，过于严密的过程并不代表有好的结果，从20世纪90年代早期到后期，靠遵从CMM获得极大成功的公司名单在不到10年时间已逐渐缩短，谈起来就像一部不断缩水的名人录。步骤过于严密对一切都在快速变化的时代不是一个正确的应对方案。提前预测你的每一步骤所应达到的目标这种有点愚蠢的方式应该结束了。在你甚至不能确信所要到达的目标是什么时，提前预计你的每一个步骤是没有什么意义的。

今天，采用臃肿过程的时代已经结束，正如Jim Highsmith所说：“采用简化过程的时代已经来临。”为了优化速度和灵敏度，必须让过程简化。抛弃日常文牍和官僚作风，简化无休止的代码检查，培养人员，使他们能驾驭自己在现代IT项目这个无序的迷宫中灵活地穿行，这些就是软件开发的敏捷方法之根本。

Jim Highsmith以概括性的方式对敏捷方法学进行了综述。他采用引人入胜的叙事方式，而不是讨论一些概念。同任何好的故事一样，人在其中起支配地位。因此，他通过给我们讲述敏捷方法的倡导者和发明者，如Kent Beck、Alistair Cockburn、Martin Fowler及其他“轻方法学家”，从而让我们了解敏捷方法。这些方法和思想正在改变着IT行业的运作方式。这就是本书所讲述的内容。

Tom DeMarco
Camden，Maine

前　言

2001 年 2 月 11～13 日，在犹他州 Wasatch 山的滑雪胜地 Snowbird 的一幢大楼中，17 个人聚在一起交谈、滑雪、休闲，试图找到某些共同话题。这次会议形成了敏捷软件开发运动，其中包括：极限编程（eXtreme Programming，XP）、Scrum、动态系统开发方法（Dynamic Systems Development Method，DSDM）、自适应软件开发（Adaptive Software Development，ASD）、Crystal 方法、特性驱动开发（Feature-Driven Development，FDD）、实用程序设计等。这些方法可满足对不同于文档驱动的、严格的软件开发过程的新方法的需要。会议的结果是 17 名与会者签署了“敏捷软件开发宣言”（The Manifesto for Agile Software Development），宣告：

我们通过实践寻找开发软件的更好方法，并帮助其他人使用这些方法。通过这一工作我们得到以下结论：

- **个体和交流**胜于过程和工具。
- **工作软件**胜于综合文档。
- **客户协作**胜于洽谈协议。
- **回应变革**胜于照计划行事。

这个观点有许多鲜明的特点，并不在于至少有 17 个人的一致同意。尽管这些人具有丰富经验并被认为是软件开发界的领军人物，但是宣言中选择词语“**寻找**”表明了作者并没有所有的答案，也不赞成“银弹理论”。

短语“通过实践”表明作者确实在自己的工作中实际实践了这些方法。Ken Schwaber 曾讲过他自己出售过自动综合的、属于“重”方法学的工具。Ken 的公司那种快速反应给 Jeff Sutherland 留下了深刻印象，他问：“哪些严格的方法学被用于内部研发?” Ken 评价道：“当我告诉他，‘从不使用，如果我们使用其中任何一个，我们就会倒闭。’我仍记得 Jeff 的脸色。”作者以敏捷开发实践来帮助他人，并在互相学习中丰富自己的知识。

上述价值陈述有一个格式。在每一陈述中，前半部分具有一种优先权，而后半部分则描述了一个项，尽管这个项很重要，但优先级要低。这种区别是敏捷方法的核心，但简单地要求别人列出什么是有价值的，还不能找到其根本差别。Roy Singham 是 ThoughtWorks 公司的 CEO，他很好地表达了这一点：正是边界情况和困难的选择使我发生了兴趣。“是的，我们评价计划、综合文档、过程和工具，这些都容易做到，困难的是问：‘哪些价值更高?’”当把这些问题提

出来时，必须有所回答，我们必须清楚保留什么，放弃什么。

第一条价值陈述认识到，过程和工具虽然重要，但人才之间的交流更重要。严格的方法学及业务过程再工程的同事们强调过程重于人。敏捷开发方法将这一点反了过来。如果我们相信个人是独一无二的，那么过程应该融入个人和团队，而不是其他什么别的方式。

类似地，虽然综合文档并非有害，但重点一定要在最终产品——工作软件上。就是说，每一个项目组必须自己决定，哪些文档在交付工作软件时是非有不可的。摆在开发者和发起人面前的工作软件说明了实质——哪些地方与当初的承诺是不同的。工作软件可以被包装、修改、分析，但却是真实的。

洽谈协议，无论是通过内部项目章程，或者通过外部的合法合同，并不是坏事，但光有这些是不够的。客户要的是交付能满足他们需求的产品。Ron Holliday 说："洽谈协议会增加不确定范围［偶然性］。这会使完成项目的时间更长，成本更高，并减缓对变化了的需求做出响应。客户和供应商的团队协作可能是最好的解决方案。"协议和项目章程提供了边界条件，这样参与者得以工作，但只有通过不断的协调，开发小组才有希望真正理解客户需求，并交付客户需要的产品。

有人能证明照计划行事是个好主意吗？没有。不管怎样，在一个业务和技术都混乱的世界里，严格照计划行事，即使是忠实地执行计划，都有可能带来可怕的后果。不管计划制定得多么详细，如果这个计划阻碍你随机应变，它就变得很危险。本书提供的一些案例研究说明，几乎没有项目是按开始计划时那样来完成的。只有那些足够敏捷的开发小组，不断适应外部的变化，才会取得成功。在信息时代，制定计划是重要的，但更加重要的是要接受任何计划都有偏差是正常现象这一观点。

2000 年春，由 Kent Beck 组织的极限编程带头人的会议在俄勒冈州召开，一些局外人，但却是"热心者"——包括我自己也应邀参会。这次会议孕育了在 Snowbird 的会议。与会者一致支持各种"轻"方法学。在 2000 年，许多论文都参考了有关"轻"或"轻量级"实践的分类。在谈话中，即使采用"敏捷"开发方法的作者们不喜欢"轻"这个词语，但在那时这个词语仍在用。

2000 年 9 月，素有"鲍勃大叔"之称的芝加哥 Object Mentor 公司的 Martin 发出一封 e-mail，从而开始了"敏捷"之球的滚动。e-mail 内容为："我希望于 2001 年 1～2 月在芝加哥举行一个为期两天的短会，会议的目的是把所有的轻量级方法的带头人聚到一起。我邀请你们所有人，我很想知道你是否能来?"鲍勃设定了一个在线论坛，从而开始了热烈讨论。Alistair Cockburn 也以书信形式加入了讨论。他对"轻"一词感到不满意："我并不在意将方法学称为'轻'，但我并不想因参加一个轻量级方法学家的会议而被称为是轻量级的，听起来有点像一帮极瘦的、低能的人努力想记住今天是什么日子。"

最激烈的争议是会议地点，芝加哥的冬天较冷，无处可玩。有人提议到犹他州的 Snowbird，虽冷，但可以玩耍，至少可以滑雪。Martin Fowler 第一天就去滑雪了。还有人提到温暖而有趣的加勒比安圭拉岛，但路途太远。最后，选择了 Snowbird 和滑雪。

敏捷宣言的价值陈述表达了一个更深的主题，这一主题激励着许多（但不是全部）宣言的签署者。在两天会议快结束时，Bob Martin 开玩笑说，他打算做一个“柔软的”陈述。尽管有点幽默，但没有人否认鲍勃的感觉。我们都感觉很荣幸能和一群有相似观点的人在一起工作，基于互相的信任和尊重，提倡以人为本的组织形式，建立工作成员之间的合作和交流。我认为关键点是，敏捷软件开发者确实具有柔软特性，交付给客户的产品可以工作在一个变化不定的场合。因此，最后分析认为，对敏捷软件开发感兴趣的人数和持批评态度的人数均有所上升，这就反映了价值和文化的柔软特性。

例如，我最后认为，XP 在应用中已经成熟，许多人对其感兴趣，原因并不是结对编程或重构，而是一种总体考虑，是在实践中定义了从公司的计划束缚中解放出来的开发团体。Kent Beck 说过以前工作中的一件事，他估计某个程序设计工作需要 2 人工作 6 周，他的经理一开始就为任务分配了另一个编程员（有效地平分资源），Kent 用了 12 周完成了任务——感到不可思议！当然，Kent 的老板在后面 6 周中大感不满，抱怨他是一个如此“失败”的程序员。Kent 最后认识到他最初关于 2 人工作 6 周的估计是十分精确的，他的“失败”其实是他的经理的失败，他没有对项目资源进行合理分配。这种情况可以说每天都发生在那些人身上：他们不愿意做出难以权衡的决策，而是强行发布不合理的命令。

敏捷运动不是反方法学的，事实上，我们在努力恢复方法学概念的可信性。我们要恢复一种平衡关系。我们接受建模，但不是为了写一些流程图后再丢进公司的储藏间；我们接受文档，但不是要数百页的从不修改和极少使用的卷宗；我们制定计划，但能认识到在一个激烈变化的环境中，计划有很大的局限性。在一些人的思想中，总把 FDD 或 Scrum 或任何一个敏捷方法归入“黑客技术”，其实这是对黑客方法和黑客基本定义无知的表现——名词“黑客”是指：热衷于解决复杂程序设计问题的编程者，而不是那些进行特定开发或解密的人。敏捷软件开发可以结合已证明的软件工程技术，但没有传统方法学的约束。

敏捷联盟诞生于 2001 年，但其各种方法和研究这些方法的人的历史可追溯到 10～15 年前。本书描述了这些人的研究实践和背后的相关原理，但更重要的是，本书剖析了正在开发各种方法的人和使用这些方法为客户创造商业价值的人。

寻找平衡点

在敏捷宣言的价值陈述中，敏捷学家认为左边项比右边项更重要，但这并不是说工具、过程、文档或合同等不重要。重要的事情和不重要的事情有极大的差别。巧妙地使用工具是绝对重要的，这能加速软件开发，并降低成本。合同是用于理解开发者和客户关系的关键文件。现代社会，文档有助于交流、增强知识传播、保留历史资料、满足政府部门和法律的要求。

但敏捷学家对这些问题有自己的看法。尝试一下，对宣言中的每一个价值陈述按以下的方

式提出两个问题：

- 我们仅交付文档而没有工作软件，项目能成功吗？
- 我们仅交付工作软件而没有任何文档，项目能成功吗？

通过考察这两个极端问题，我们就能更好地领会相关问题的重要性。尽管文档和工作软件、合同和协作、灵活反应和计划、人员和过程都必须有一个平衡点，我们还必须描述极端、终点情况，以使组织、团队和个人能找到自己的平衡点。如果一开始就想找中间状态，我们只会落空。

XP 的“巨头”Kent Beck、Ron Jeffries 和 Ward Cunningham 的伟大贡献之一是把自己立于标杆位置而引起各种争论，如果位于调和位置则不会发生这种情况。当 Ron 说“伟大的设计来自于空白的先行设计，并需不断重构”时，他是让自己，也是让我们重新思考关于软件开发的假设。我们必须理解各种局限性，然后才能理解平衡点。

所以，尽管我认识到文档、合同、计划和过程的价值，但有关这些主题已有大量的资料。我的目的是识别和定义敏捷软件开发，讲述其实践和原理。因此，你可以自己决定，哪些知识和内容是你或你的组织所需要的。

基本问题

本书回答 3 个基本问题：

(1) 敏捷方法最适合于解决哪些问题？

(2) 什么是敏捷？

(3) 什么是敏捷软件开发生态系统（Agile Software Development Ecosystems，ASDE)？

敏捷方法最适合于解决哪些问题

敏捷方法最适合于解决那些具有快速变化、动荡无序性质的问题。有些人争辩说，好实践就是好实践（例如结对编程、客户重点小组或特性规划等），因此 ASDE 不应限制于那些特别的问题类型。从部分来看，此话不假。上述问题是问，哪些问题用敏捷方法解决得最好，而不仅仅是能解决哪些问题。因此，尽管 XP、Crystal 或 Scrum 方法确实能应用于范围广泛的项目，但它们特别适于极端的或复杂的项目，这些项目有一个不断加速的开发时间进度表，且在项目完成过程中会产生持续变化而具有很大的风险性和不确定性。

随着由技术、业务模型和产品的快速演变而引起的整个变化层次的提高以及对加快产品交付速度的需求增加，ASDE 的有效性会很快超过严格的方法学。

什么是敏捷

同任何其他复杂概念一样，敏捷也有许多定义，但我最注重的定义是：

敏捷是为了在动荡的业务环境中获益而创造变革和响应变革的能力。

敏捷组织没有在变革中萎缩，而是利用或接受这种变革，这样既能比竞争对手更好地响应多变的环境，又能创造使对手无法应付的变化。然而，各公司必须决定为保留竞争力所需要的敏捷程度。敏捷仅仅是一个和竞争对手相关的优势——一个铜矿公司没有必要像生物技术公司那样敏捷。

敏捷的其他方面也很重要：既要灵巧或灵活，又要平衡。敏捷组织是灵巧的（能够迅速改变方向）、灵活的（能明白上周还是正常工作的事情，下周可能就不是了），敏捷组织也知道如何在结构化和灵活性之间取得平衡。如果所有事物一直在变，就难以前行。敏捷组织认为在有序和无序之间找到平衡点是成功的关键。

什么是敏捷软件开发生态系统

我从开始写这本有关敏捷软件开发方法学的书以来，就一直对“方法学”这一词语感到担心，因为它并不适合于描述敏捷开发的关键点：人才、关系和不确定性。而且，如果使用词语“方法学”，人们就会把敏捷实践和传统的软件开发方法学进行比较，并在比较中使用错误的测量尺度。所以，我使用“敏捷软件开发生态系统”来描述有3个成分交织在一起的整体环境。这3个成分是：混沌有序的观点、协作的价值和原理，以及刚好够用的方法学；并用“敏捷学家”来称呼那些ASDE的支持者。

某些人认为“敏捷”就是较少过程、较少形式和更简洁的文档。其实敏捷还有很多特色，这是使用词语生态系统而不是方法学的主要原因。尽管较少过程和形式可以降低开发成本，但这些还不足以产生敏捷。将重点集中在人才及相互作用，给予个人快速决策和自我适应其过程的权力是敏捷生态系统的关键。

《美国传统词典》（American Heritage Dictionary）将生态系统定义为“有机体及其环境：局部团体在一定环境中互相依赖生存《牛津英语词典》（Oxford English Dictionary）把这一定义进行扩展，包括了系统内的不断交换，既有有机的元素，也有无机的元素。词语方法学涉及很多事物——活动、过程、工具。这些并不是无关的元素，但并不完善。而词语生态系统则涉及很多活的事物以及相互之间的交互。在一个组织的环境中，一个生态系统被认为是一个动态的、不断变化的环境，在里面的人和组织不断有新行为，也对别人的行为做出反应。生态系统的关键是个体和团队的动态交互，而不是组织图中静态排列的一行行图符。

混沌有序的观点

为充分理解 ASDE，我们需要理解 3 个成分中的每一个及其之间的相互关系。首先，敏捷学家有一个共同观点，即组织是混沌有序的，每一个组织的动荡和有序表现在与使用线性的、可预测的计划和实施唱反调。具有了组织是混沌有序的观点，就能理解：传统项目管理和开发生命周期实践所依赖的可预测性是导致客户、管理和开发组织之间不协调的根本原因。混沌有序的观点既影响我们如何响应变革，也影响我们如何管理项目团队和组织。

在日复一日的项目工作中，混沌有序的观点产生了两个结果，这两个结果和严格方法学是根本对立的。

- 产品目标是可以达到的，但它们是不可预测的。
- 过程能够帮助维护一致性，但它们是不可重复的。

尽管 ASDE 也包含详细的计划，但基本假设却是：在一个动荡的环境中，计划是不可预测的，至少在项目的范围、时间安排和成本层次上是这样。计划是需要检验的假设而不是要实现的预想。然而，业务的产品目标是可以达到的，这主要是因为敏捷人员的适应能力。通过权衡各个方面，可以使业务的产品目标适应相互联系的构想、项目的时间安排、范围或成本目标。第二，虽然过程能帮助人们一起工作，但在一个不稳定的环境中，通过测量和修正（静态过程控制）来排除过程的变化的想法变成了不适宜的假设。在完成项目过程中，不断获得知识就会产生变革，有些知识在项目的早期是察觉不到的，这需要过程能响应变革，而不能试图忽略它。

正如 Martin Fowler 指出，ASDE 有两个基本特性，第一是强调可适应性，而不是可预测性；第二是强调人，而不是过程（Fowler 2000）。通过阅读本书，读者会弄明白这两条特性的基础地位和对现状的挑战性。成为敏捷意味着接受了这样的观点：结果不可预测且过程不可重复。甚至于可以说，随着过程的可重复性增加，项目的可预测性会减少。

Peter Senge 使用术语“智力模型”来表示我们每个人的观点、各种假设、故事和信念，它们构成了人的思考空间（Senge 1990）。在组织中，智力模型的集合就确定了整个组织的文化氛围。面向销售的公司不同于面向项目的公司。以客户满意为战略的公司不同于以产品创新为战略的公司。其智力模型中包含了直线性、因果关系、等级、可预测性和控制的公司与那些在智力模型中包含协作网络、紧急情况、权力分散并接受不可预测性的公司的运作是大不一样的。前者按牛顿学说，而后者是混沌有序的。

混沌有序的观点构画出了一个复杂的自适应的系统模型，其分散的、独立的智能体（每一个人）在一组简单的但能生成复杂结果的再生规则指导下，以自我组织的方式进行交互。这一

观点在我的著作《自适应软件开发》(Adaptive Software Development)(Highsmith 2000)中进行了详述，并且明确包含敏捷学家 Ken Schwaber、Bob Charette 和 Kent Beck 的哲学思想。

XP 提供了一个方法学和文化如何相融合的例子。一方面，XP 被定义为一个实践的系统：结对编程、测试优先的开发、重构、编码标准化。另一方面，XP 的价值和原理定义了协作的文化——开发者如何同客户一块工作、人与人之间如何交互和相处。

协作的价值和原理

在互连的 Web 上定义 ASDE 的第二个方面是协作的价值和原理。虽然用一个词语来全面描述敏捷宣言是很困难的，但是“协作”可能是最好的一个词。价值和原理塑造了生态系统。如果没有一组可说明的价值和原理，生态系统就是枯燥无味的，反映了人的工作实践，但没有反映在其中相互作用的人。

协作文化既包含了人，也包含了一个开发团队中人与人之间的相互关系，以及开发团队中的成员与客户、与管理者、与公司内外的合作团队之间的相互关系。人类的能动性、交流和协作可以被称为“软”科学，但实践中它们却是最难驾驭的科学。原理和价值有助于定义一种文化，即人们工作的一种环境。

刚好够用的方法学

ASDE 的最后一个组件是方法学。方法学的传统定义包括角色、活动、过程、技术和工具。Alistair Cockburn 将方法学定义为“我们都同意的公约”——据此，人们在一个项目组中一起工作。John Seely Brown and Paul Duguid (2000) 在《信息的社会生活》(The Social Life of Information) 一书中，讨论了（由业务过程再工程运动所使用的）过程和实践的主要差别。过程以手册来描述，实践则是实际发生的事情。过程中心论者将人置于第二位，实践中心论者将人置于首位。过程中心论者将注意力放在明显的（已经写下来的）知识上，而实践中心论者将注意力集中在不用明说的（内在的）知识上。ASDE 模型给方法学提供了一种以实践为中心而不是以过程为中心的途径。

有两个理由可以说明刚好够用的方法学：价值和创新。精简了的方法学只注意那些创造价值的活动，而无情地忽略没有增值的活动。编写程序通常增加价值；过程管理则增加企业管理费用。刚刚够用意味着保留前者而丢弃后者。第二，创新和创造性在混沌有序的环境中都很活跃，而在有序的环境中则不然。刚好够用的方法学是孕育创新的摇篮。

方法学也同组织模式有关。敏捷方法学包含了最少的过程和文档，尽量简化的形式。敏捷方法学的标签是“刚好够用”(Cockburn) 或“略少于够用”(Highsmith) 或“最小化”(Charette)。

然而，这一精简了的方法学并不减少工作效果，更重要的是，基于对混沌有序世界观的理解：创新的结果出现在“混沌边缘”，处于混沌和有序的中间地带。

实践（或技术）是方法学的命脉。无论是结对编程、Scrum 会议、客户重点小组，还是自动测试，这些由杰出的专业人员进行的 ASDE 实践都会带来丰富成果。

变化的观点

在软件专业，我们已经使用“方法学”和“过程”这些词语很长时间了，不论是在口语中还是在书面用语中均没有问题。“生态系统”这个词语也会逐步得到使用。我使用“生态系统”这一词语的目的是为了培育不同的观点。Arie De Geus（1997）曾说过：“越来越多的证据说明，公司的失败主要是因为其管理层的主导思想和语言过于局限于当时在经济学领域占主导地位的思想和语言，而忽视了其组织的真实本质是人类的团体性。”

为了改变思想，我们必须改变使用的语言，所以我使用“生态系统”来改变我们对软件项目的看法。**混沌有序的观点、价值和原理的协作集**以及**刚好够用的方法学**，三者相互结合，共同作用，形成了敏捷生态系统。至少在我的想法中，不能将三者分开，我想大多数敏捷学家也持同样的观点。你可能具有精简了的方法学，但用线性的、牛顿式的组织观念，所产生的结果不是敏捷的；你可能具有精简了的方法学，但用有层次的、基于控制的工作文化，所产生的结果也不是敏捷的；你可能具有协作的、以人为本的工作文化，但采用严格的、可预测的方法来计划和管理项目，所产生的结果仍不是敏捷的。敏捷方法需要所有的三方面共同作用。

本书的根本目标是描述新的工作方式，以便向软件客户提供有商业价值的产品。ASDE 的精髓是对人的信任——信任他们的个人及其交互行为。讨论人及其合作工作方式（生态系统）就要讨论价值和原理；讨论价值和原理就要讨论一个组织是如何工作的或应该如何工作；仅仅使用“方法学”作为惟一的比较机制是不可能将敏捷方法同非敏捷方法进行比较的。

最后，我要声明，我不代表敏捷团体中的任何其他人，只代表我自己。我可能解释了 Ken Schwaber 有关 Scrum 的观点、Jeff De Luca 有关 FDD 的观点、Alistair Cockburn 有关 Crystal 方法的观点，但那只是我的理解。在对 ASDE 的一般介绍中，我相信签署敏捷宣言的其余 16 个人中的一个或更多的人会不同意我的某些陈述。然而，我和他们曾经交谈过或一起工作过，因此尽管那些陈述虽然是我自己的理解，但也反映了敏捷团体的一部分人及相关研究人员的深层认识。

尽管只有 17 个人签署了敏捷宣言，但有成千上万人支持这个宣言。许多人已在宣言网页上签名，网站上的大量留言是“我们支持敏捷宣言”。敏捷学家已经在软件开发和项目管理界发起了有积极意义的争论。我希望本书能有助于这种争论，鼓励其他人参加进来。

Jim Highsmith

2002 年 1 月于犹他州盐湖城

引　言

本书由四部分组成。

第一部分阐述变革驱动的信息时代经济的关键特征，讨论传统的严格软件开发方法为什么在这一环境中难以取得成功。第1章描述现在动荡的经济条件，并说明在将来这种动荡不会减弱。第2章包括IDX系统公司的一些事例，说明一个处于混乱状态中的公司会面对的问题，也说明了使用敏捷软件开发生态系统能达到的结果。第3章以宏伟画笔勾画出由加速变化引起的问题的解决方案：敏捷方法。

第二部分探究ASDE、人、作者以及核心思想领导人（他们发明了ASDE）的价值和原理。在有关原理的各章中，交错讨论了那些表达原理的人。还有什么方式能比直接同核心人物进行讨论更有助于理解敏捷方法的价值和原理吗？各章并没有严格按敏捷宣言的原理进行安排，而是按我自己的分类安排。这种访谈没有直接解释每一个敏捷方法，但表达了他们如何以个人经验来理解软件开发。

深入下去，读者不必对Kent Beck、Alistair Cockburn、Ken Schwaber或其他敏捷带头人有很深印象，就会发现有一些人致力于把软件开发和信息技术的世界变成令人满意和愉快的工作场所。不理解创始人关于工作关系的观点及其内在的文化价值观，就不能真正理解敏捷运动。同他们会谈的内容也在各章中，以帮助读者加深理解。

同样在第二部分，有关原理的每一章中都以一个事例开头，每一个事例来自于将某个ASDE成功应用于一个项目的组织。这些例子覆盖了广泛的项目类型，来自于新西兰、印度、新加坡、加拿大、美国和德国等世界各地的国家。

第三部分介绍各个敏捷软件开发生态系统。每章描述一个敏捷方法的基础及关键贡献。内容涉及Scrum方法、动态系统开发方法、Crystal方法、特性驱动开发、精益开发、极限编程和自适应软件开发。

第四部分讨论一个组织如何对其文化背景进行分析以决定一个合适的ASDE如何为自己设计和定制一个方法学框架及实践。

本书的组织和约定

在组织本书内容时，总会有许多东西难以取舍。其中之一是敏捷方法中的哪一个放在前

面，或者是否让有关原理的几章优先。我选择了后者，因为最终我相信价值和原理比实践更重要。如果读者爱好新奇，就跳过开头的一章或几章，首先阅读有关 ASDE 的几章。在每章的引言中，都有一段文字介绍 ASDE 及主要开发者。

贯穿此书，有一些约定。首先，我认为每一个软件开发项目都是一个产品开发项目。我多年和一些为内部客户开发软件的 IT 组织，以及开发市场和销售软件的软件公司一起工作。在以上两种情况下，开发人员都要向客户交付产品。牢记客户和产品不仅有助于产品开发公司也有助于 IT 组织。

第二，我所使用的开发者一词是指直接参与向客户交付产品的所有人员，包括程序员、测试员、文档专家、需求分析员以及其他有关人员。在我每次提到交付工作软件的人员时，并不想提 4 或 5 种角色。尽管交付工作软件是敏捷开发的关键价值，但编程并非完成目标所要求的惟一角色。

在我谈及软件开发或软件开发实践时，通常还在句子中包含有项目管理或协作实践的意思。若在本书中一直使用短语“软件开发、项目管理和协作”这样的形式，则在讲任何其他有用的内容前已经占用了半个句子，故我经常将其缩写，希望不会引起误解。

经审慎考虑，我最后决定用“严格的”来称呼非敏捷方法，部分原因是因为这个词并不十分绝对。敏捷意味着结构和灵活性之间的平衡，所以严格是任何开发过程的关键部分。敏捷方法主要关注定义的灵活性方面，而严格方法则主要关注结构方面，故我用这两个词来说明开发的两种截然不同的风格。太多的结构会使严格变成呆板。同样地，过于灵活会使敏捷变成混沌。我用敏捷和严格作为对比，以强调各自的基本特征。尽管我努力想中立地来看这些词语的用法，但对敏捷的偏爱还是比较明显。

为了不是总在书中使用敏捷生态支持者或敏捷方法学家这样的词语，我在此运用“敏捷学家”一词。尽管这种用法并不完美，但使用这个重复性很高的词可以使句子变得简洁。

最后，本书中的所有例子均是我的实践或与我交谈过的人的实践。有些公司或个人不愿被提及，所以我在相关地方使用了化名。对所有提供了素材但未指明的人或公司，深表谢意。

主要的敏捷生态系统及其带头人

本节对每个主要 ASDE 及其主要开发者进行介绍，并简要说明各个方法。

Scrum

Scrum，在橄榄球赛中是并列争球的意思，最初由 Ken Schwaber 和 Jeff Sutherland 提出，后来 Mike Beedle 也参与协作。Scrum 提供了一个项目管理的框架，其核心是一个 30 天的 Sprint 周期，在这周期内一系列的产品特性得以实现。Scrum 的核心实践是每天用 15 分钟时间举行团体

会议进行协调和集成。Scrum 方法已经使用了近 10 年，并成功开发了范围广泛的产品。第 8 章包括同 Ken Schwaber 的交谈，第 17 章描述 Scrum。Ken、Jeff 和 Mike 均是敏捷宣言的签署者。

动态系统开发方法（DSDM）

动态系统开发方法于 20 世纪 90 年代中期在英国提出。它是快速应用开发（RAD）实践的派生和扩展。DSDM 至少在欧洲是 ASDE 中对培训和文档支持得最好的。DSDM 的 9 个原理包括：积极的用户参与、频繁交付、团队决策、在整个项目生命周期中的集成测试以及开发过程中的可恢复改变。有关 DSDM 的描述、与 Arie van Bennekum（DSDM 联合会理事会成员，敏捷宣言的签署者）的谈话内容，可以参见第 18 章。

Crystal 方法

Alistair Cockburn 是以人为本的方法中的 Crystal 族的作者。Alistair 是一个“方法学的考古学家”，他同全球几十个项目开发团队进行访谈，试图把实际起作用的方法学和认为应该起作用的方法学区分开来。Alistair 及其 Crystal 方法主要关注开发过程中的人之间的协作、良好关系和合作。他使用项目的规模、重要程度和目标来精心并适当地描述方法学中每个 Crystal 方法的实践。第 6 章包括同 Alistair 的交谈，第 19 章描述 Crystal 方法。Alistair 也是敏捷宣言的一位签署者。

特性驱动开发（FDD）

Jeff De Luca 和 Peter Coad 协作提出 FDD。FDD 由一个最低限度的 5 步过程组成，其核心是开发一个完全“定形”的对象模型、构建一个特性列表、然后按特性来规划、迭代进行按特性的设计和按特性的构建。FDD 的过程很简明（每一过程可在一张纸上予以说明），它的两个关键角色是首席构架师和首席程序员。FDD 与 XP 的不同点在于其“轻”的前端体系结构建模方法。在第 20 章，澳大利亚人 Jeff 提供了使用 FDD 的大型项目（超过 50 人）的两个实例。Peter Coad 的公司——TogetherSoft 的产品研发部经理 Jon Kern 也是敏捷宣言的签署者。

精益开发（LD）

对 ASDE 最具战略意义的也最不出名的是 Bob Charette 的精益开发方法，它是从 20 世纪 80 年代日本汽车制造业大调整的精益生产原理中演变而来。在 LD 中，Bob 扩展了传统方法学所持的用严格的管理实践来控制变革损失的观点，认为变革会为“风险企业家”带来求之不得的机会。LD 已成功应用在许多欧洲的大型电信项目上。同 Bob 的交谈请参见第 16 章，第 21 章描