第 2 章　敏捷软件开发

每天早晨起床，我既想改变世界，又想玩儿个痛快。有些时候，这让我很难好好地规划当天的工作。

——E. B.怀特，美国作家(1899—1985)，代表作品有《夏洛的网》和《精灵鼠小弟》等

本章对于部分读者而言是可选的。如果你熟悉敏捷软件开发，想必已经比较熟悉本章的大多数内容。本章旨在提供敏捷软件开发的概况，特别针对希望在我们开始探讨敏捷组织中的管理角色(参见第 4 章)之前进一步了解敏捷的背景和基础的读者。

本书自始自终，我都假设你对敏捷软件开发的基础知识略知一二。但在这里，先假设你认为极限编程已经过时，并想继续阅读下文。

敏捷前传

对我而言，算钱和花钱一样其乐无穷。上世纪九十年代初，当时我还在荷兰 Delft 科技大学读书，利用课余时间，我写了一个账本程序(参见图 2.1)。我之所以这样做完全是出于个人兴趣，然而美中不足的是，我那时并没有什么钱可以算。但内心的某个阴暗角落还是满怀希望地想，只要自己想算，百万美金自然会滚滚而来。但是，哎，这想法一直也没有实现。

我一个人完成了整个程序，大概写了 3 万多行代码。我没有正规的方法，没有软件开发经验，没有经理、教练和顾问的帮助。但是我有的是时间、电脑、愿景和完成伟大产品的强烈动机。

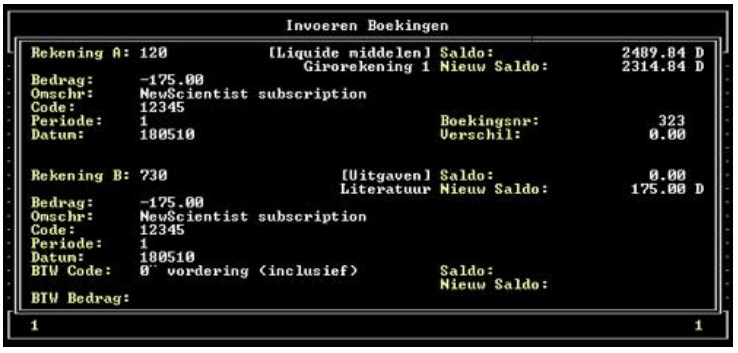


图 2.1　JEBS 2.0，我 20 多年前写的账本程序(荷兰语)

令人称奇的是，这个软件我竟然卖出去几十份，有些客户甚至被完全震惊，他们没想到账本程序也可以如此简单、操作友好并且好看(就上世纪九十年代而言)。现在，二十多年过去了，我仍然在使用这个老掉牙的程序记账。在这二十多年的使用过程中，我只发现了三个小问题。

这怎么可能呢？一个没有经验的程序员是如何构建出如此高质量的软件，并且用了几乎近二十多年毫无差错的呢？

对此，我完全不知道为什么。

但是……我可以列举敏捷专家看着觉得亲切的下面这些情形。

- **我对构建产品充满了激情。**我有账本应用程序的使用经验，确信这些应用操作性很差，都是来自地狱的数字宠儿，不但榨干了用户的生命，而且每次敲击键盘都能使用户灵魂出窍。我的愿景就是要让程序与众不同。不像别的同类软件，我的程序操作友好，使用愉快。

- **我本人就是一个关键客户。**我不是为别人而是为自己写程序。当然，我很高兴能找到客户，即使这些客户没能让我成为内心深处所渴望的百万富翁。但是我做了一切所能做的工作，确保程序以我想要的方式运行。

- **我没有计划，只有一个功能列表。**首先，我从核心功能入手，比如输入新的交易。接下来是一些非核心功能，比如余额调整和差错更正。最后处理一些

锦上添花的功能，比如帮助页面或导出功能。直到我厌倦了这一切，就简单地宣布产品已经准备就绪。

- **我在构建产品时不断地完善流程。**我按照一个简单的清单往下开发。这个清单记录了每个步骤，而且随着时间的推移，清单稳步增长。虽然我从来没有听说过单元测试，但我日常的行为准则完全可以和飞行员媲美，会一而再、再而三的检查，确保产品质量没有问题。

就是这样。我有强烈的动机，有关键客户，没有前期的计划，遵守严格的纪律和自组织的流程。即使我以前从来没有开发过账本应用也没有关系，真正重要的是我渴望学习。

在我完成账本应用十年之后，我发现我以前遵循的(部分)流程，现在突然被称为“敏捷软件开发”。现在，经过十年的学习，我着手写一本书来介绍敏捷所缺失的一种成分。这本书涉及的范围和我那个账本程序差不多。当然，嘿嘿，和以前一样，我内心深处那个阴暗的角落已经蠢蠢欲动，准备算钱了。

敏捷之书

起初，工程师创造电脑及软件。软件混沌而糟糕，用户脸色阴暗。工程师说“要结构化”，于是就有了结构化。[①]

实际上，结构化得有些过分了。

在过去的五六十年，许多软件工程师一直关注着软件质量的巨大起伏，这些软件质量的好坏取决于软件开发人员所采用的即兴开发方法。于是，他们着手创世，并最终得到了许多正规方法。软件工程专业[②]应运而生。软件工程认为软件开发是一种工程活动。由此，它引入了许多模型、方法、框架、语言、模式和技术，

① 译者注：作者这里模仿了《圣经》创世纪

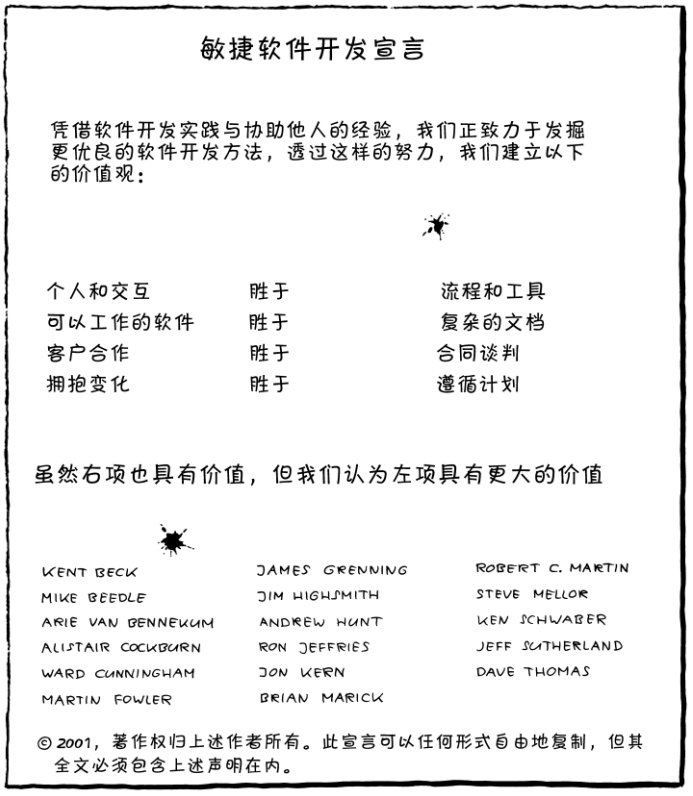
② http://www.mgt30.com/software-engineering/

以期帮助程序员制造出更好的软件。但奇怪的是，多数项目仍旧不尽人意。而且更多的时候，这些正规方法徒生了官僚主义。通常，软件产品开发周期很长，并且需要大量的文书工作，以至于那些“确定的”需求在系统交付之前早已经发生了变更。与此同时，反而是一些由充满激情和遵守纪律的程序员所组成的小团队交付了高质量的产品。即使只有些特设的流程，并且需求在不断变化，他们仍以较低的成本和较短的时间完成了任务。创造过程虽然产生了恐龙，然而小小的蚂蚁却能满载而归。

上世纪九十年代初，“快速应用开发”(RAD)[①]方法诞生了。这种方法结合了重量级软件工程的一些正式的技术(比如变更委员会、检查和度量的标准)和来自许多成功项目团队的切实实践(比如原型、阶段性交付、频繁的客户合作)[McConnell 1996]。这种正规与特定开发方法的联姻繁荣了一系列以“轻量级”命名的软件开发方法，它们包括 Evo[②](1988)、Scrum[③](1995)、DSDM[④](1995)、水晶方法(Crystal)[⑤](1997)、极限编程(XP)[⑥](1999)、特性驱动开发(FDD)[⑦](1999)、实效编程[⑧](1999)以及自适应软件开发[⑨](2000)。

如同寒武纪生命大爆发，一时之间，针对轻量级软件开发的方法、文章、图书和研讨会如雨后春笋般大量涌现。因此，有些专家希望能将这场运动的领导者组织起来开一次会。2001 年，他们聚集于美国犹他州的一个滑雪胜地。在这次聚会上，他们决定用“敏捷”来替代“轻量级”，由此，敏捷宣言[⑩]横空出世(参见图 2.2)。

① http://www.mgt30.com/rad/
② http://www.mgt30.com/evo/
③ http://www.mgt30.com/scrum/
④ http://www.mgt30.com/dsdm/
⑤ http://www.mgt30.com/crystal/
⑥ http://www.mgt30.com/xp/
⑦ http://www.mgt30.com/fdd/
⑧ http://www.mgt30.com/prag/
⑨ http://www.mgt30.com/asd/
⑩ http://www.mgt30.com/manifesto/



敏捷软件开发宣言

凭借软件开发实践与协助他人的经验，我们正致力于发掘更优良的软件开发方法，透过这样的努力，我们建立以下的价值观：

个人和交互	胜于	流程和工具
可以工作的软件	胜于	复杂的文档
客户合作	胜于	合同谈判
拥抱变化	胜于	遵循计划

虽然右项也具有价值，但我们认为左项具有更大的价值

KENT BECK	JAMES GRENNING	ROBERT C. MARTIN
MIKE BEEDLE	JIM HIGHSMITH	STEVE MELLOR
ARIE VAN BENNEKUM	ANDREW HUNT	KEN SCHWABER
ALISTAIR COCKBURN	RON JEFFRIES	JEFF SUTHERLAND
WARD CUNNINGHAM	JON KERN	DAVE THOMAS
MARTIN FOWLER	BRIAN MARICK	

图 2.2　敏捷软件开发宣言

很多人认为，敏捷宣言是对那些正式的、按部就班的官僚化方法的反击。然而，几乎没有人意识到它也反对软件开发世界中那些不和谐的地方，比如无纪律性的程序员，混乱的流程，低劣的质量产品。这场新运动的领军人物在结构化和非结构化、秩序和混乱之间发现了一条中间路线。从某种程度上讲，这是一次英勇无畏的尝试，满心希望能够回归早期激情澎湃的开拓岁月，摈除因无政府状态所引起的惯常的种种恶行。

一些敏捷大师随后组建**敏捷联盟**[①]。这是一个非盈利性组织，旨在向全球推广敏捷软件开发。而后，一个汇集会议、咨询、图书、杂志的新兴生态系统诞生了。

① 敏捷联盟有自己的官方网站，网址为 http://www.mgt30.com/agilealliance/

软件开发开始变得敏捷，Agile 以大写的 A 开头，象征着它比一整套的软件开发实践更深刻。就如同芸芸众生，项目通过探索和确认在秩序和混乱之间生存下来，而敏捷成为一种生活方式(Way of Life)。

敏捷的基本原则

如今，敏捷实践者(Agilist，坚守敏捷价值观和原则的人)已经突破百万。调查结果显示，全球大部分软件工程师都至少尝试过一些“核心敏捷实践”[VersionOne 2009]。

敏捷的基本原则已被阐述过无数次，而且许多作者都比我诠释得好。但我仍然觉得有必要在书中包含一个简短的介绍。作为一名敏捷专家，我倾向于按自己的方式来做事，所以我将用自己的“软件项目的七个维度”来表述各个敏捷要素。我将在第 11 章重拾这个主题。

人

首先最重要的是，敏捷认为人是独一无二的个体，而不是可替换的资源，同时人的最大价值不在于他们的头脑，而在于交互和合作。敏捷要求不同角色(开发、设计、测试和其他)的人员组成跨职能的小团队，并且最好处于同一个房间内。这些团队是自组织的，这意味着没有任何强加于他们的方法和过程。团队获得信任从而可以用他们认为最好的方法去完成工作——前提是他们知道怎么做，并对结果负责。

职能

敏捷宣言明确指出，客户只有与产品开发团队一起工作才能创建出最好的产品。团队和客户(或客户代表)一起合作，维护一个始终变化的未完成功能列表，并持续为其设定优先级。这些功能的描述简明扼要，只有在团队着手处理时，这些功能才需要更全面的探索和文档化工作。对所有功能来说，简单是良好设计的关键，

当功能实现后，客户将立即验证其有效性。

质量

关注质量是产品取得成功的关键，所以技术上的精益求精在敏捷核心中找到了自己的一席之地。我们可以借助下列实践达到目的：测试驱动开发[①](写产品代码之前先写测试代码)；代码审查(可以通过结对编程来实现)；定义完成的标准(检查清单)；迭代开发(由于新的变化或理解而改写代码)；重构(持续改进代码，即使没有需求变化的时候)。敏捷专家承认**涌现式设计**的必要性，即最好的架构不是预先就定义好的(可能有一个基本的形式)，它可以在产品开发过程中逐步浮现出来。

工具

敏捷专家认为，工具对产品成功的贡献微乎其微，然而敏捷文献中却充斥着对工具的描述和宣传。经验丰富的敏捷团队更喜欢使用工具来辅助进行每日构建、持续集成和自动化测试。敏捷软件开发需要团队自我激励。但重复性的劳动不仅枯燥无味，还会使人缺乏动力，因此这些工作需要自动化完成。很多敏捷专家也要求环境的支持，比如开放的办公环境，信息透明化工具，比如大的任务板和燃尽图。在敏捷环境中，工具的目的是促进团队的激励、交流与合作。

时间

敏捷和时间有着特殊的关系。对于敏捷项目，与预算一样，交付日期和最后期限几乎可以任意选择。软件以较短的时间期限进行构建，通常是时间盒或 Sprint，并以增量的方式进行发布，而每个发布都是一个潜在可交付的产品。这就确保了业务负责人可以依据其对功能的需要和时间的要求来进行时间安排，前后调整发布日期。同时，团队总是孜孜以求可持续的开发节奏，使其能够长期保持其开发

① http://www.mgt30.com/tdd/

速度。

价值

敏捷宣言诞生的主要原因之一是为了拥抱变化。环境从来都不是静态的。昨天还价值连城的功能或许一夜之间就变得一无是处，即使已经交付给用户的功能也不能幸免。敏捷专家尝试通过缩短反馈和交付周期来应对挑战。频繁的产品发布不仅意味着收集反馈，还意味着把这些发现反馈到开发过程中。更重要的是，一旦发现新的需求，可以及时向用户交付新的功能和调整后的功能，从而提高其商业价值。

流程

尽管敏捷宣言提倡人胜于流程的模式，但这并不意味着流程不重要。更为重要的是，敏捷环境本身就包含许多必要的流程：最小计划(或递进式规划)，每天面对面的交流(通常是站立会议的形式)，通过评估可以工作的软件来衡量项目进度(客户签收的功能)。敏捷专家也承认持续改进的必要性。据此，定期的回顾和反思可以不断地评估和调整过程本身。

冲突

以上便是我心目中的敏捷基本原则。当然，它们只是我个人的想法。一些敏捷专家或许并不认同我写的这些简短介绍。但这也是敏捷的一部分，我甚至想把“冲突”列为敏捷软件开发的第八个维度。就像你随后将要看到的一样，内部冲突既是复杂系统固有的一个方面，同时也是创新的土壤。对于那些乐于持续改进的人来说，它将是一种巨大的特权。

敏捷的竞争

没有毫无竞争的比赛，也不存在毫无冲突的系统。如果没有反对意见，那我们的

世界将了无意趣。幸运的是，在敏捷的世界里有很多良性的竞争，比如 Scrum 与极限编程，Scrum 与看板(Kanban)，甚至 Scrum[①]与 Scrum[②]之间。然而，各式各样的敏捷方法并不是这场竞赛中唯一的选手，还有几个很有前途的、强劲的竞争对手贡献了各自的想法，这些想法有时是相似的，有时是互补的，有时又是绝对矛盾的。

较大的竞争对手之一是**精益软件开发**[③]。精益软件开发是指将精益生产的概念应用于软件开发的领域。精益的七个原则[Poppendieck 2009:193]以在丰田方法[④](丰田公司的管理理念)的 14 个原则和戴明[⑤]的 14 个管理要点为基础。精益和敏捷有明显的共通之处，这就是为什么精益和敏捷经常会有相同的专家从事相同的领域，分享相同的信众，出现在同样的博客、杂志和电视节目中。精益软件开发旗帜鲜明地集中于消除浪费和整体优化，从管理学的角度对敏捷世界做出了巨大的贡献。虽然精益方法较敏捷晚几年加入软件开发联盟，但精益运动通过发展自己的会议、顾问、教练和协会，已逐步迎头赶上。[⑥]

软件工艺运动是一个规模略小却颇具实力的竞争对手，其指导思想是软件工艺宣言[⑦](参见图 2.3)，该宣言既是对原有敏捷宣言的扩展，也是对它的挑战。软件工艺的支持者认为软件开发者不是工程师，而是工匠(一些人以中世纪欧洲学徒模式作比喻)。工艺运动通过自己的(小型)活动、图书和论坛，正成为敏捷和精益共同面对的一个灵活而无畏的新对手。由于同处于轻量级软件开发阵营，所以尽管它们偶有冲突，但这三者似乎正在汇合成一个伟大的团队。

① Scrum 联盟也有自己的网站，官方网站 http://www.mgt30.com/scrumalliance/
② Scrum.org 是由 Scrum 之父 Ken Schwaber 创办的，官方网站 http://www.mgt30.com/scrumorg/
③ http://www.mgt30.com/lean/
④ http://www.mgt30.com/toyota/
⑤ http://www.mgt30.com/deming/
⑥ http://www.mgt30.com/leanssc/
⑦ http://www.mgt30.com/craftsmanship/

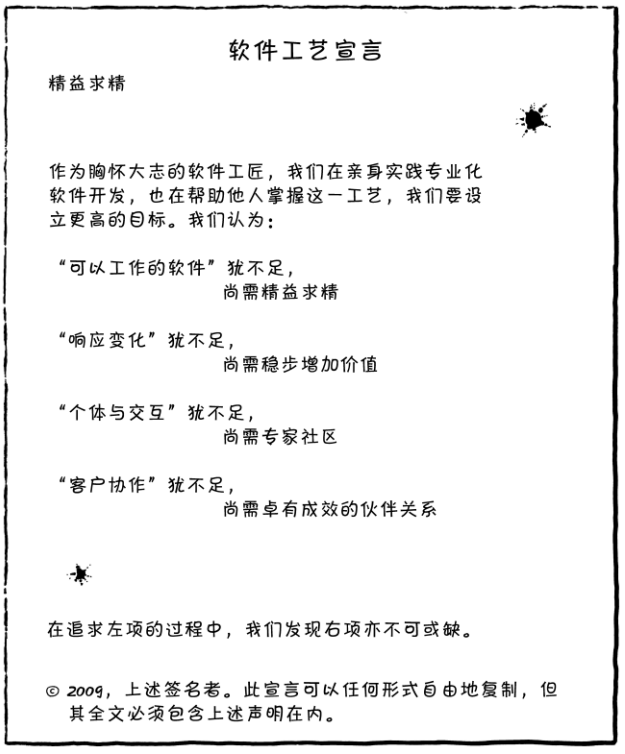


图 2.3　软件工艺宣言

当然，重量级方法和框架也没有停滞不前。这其中最著名也最具争议的就是**能力成熟度模型集成** CMMI[①]。CMMI 始于 1987 年，由卡耐基 • 梅隆大学软件工程研究所(SEI)开发和维护。起初，CMMI 充当的是软件工程过程改进的描述，但它已逐渐发展成为一个更加抽象的框架，目前其涵盖除软件开发以外的其他许多行业。CMMI 这一方法学旨在通过描述 5 个成熟度级别和 22 个过程域来指导软件开发。CMMI 只能指出在过程改进的工作中可以解决哪些过程域的问题，但没有规定如何实现。正是由于这个原因，一些敏捷实践者相信 CMMI 是可以和敏捷软件开发相互兼容的——尽管 CMMI 有着长达数百页的描述，因为敏捷方法可以补充 CMMI 所缺失的过程实现部分。然而敏捷实践者倘若不彼此反对的话，

① http://www.mgt30.com/cmmi/

他们就不是敏捷实践者了。因此，有些人认为，尽管 CMMI 出发点是好的，但是 CMMI 相当繁琐，以至于它不但把机构推向官僚化，还使团队陷入金玉其表而败絮其中的境地。

由项目管理学会发布和维护的项目管理知识指南[①](PMBOK)也不乏类似的矛盾信号。有趣的是，指南最初旨在介绍项目管理通用的最佳实践。但自 1987 年第 1 版之后，经过若干次修订，该指南已变得越来越“敏捷”了，以回应敏捷项目经理所取得的成果。和 CMMI 相反，对于很多流程，PMBOK 具体指导项目经理的工作。但是这些推荐的实践并非总与敏捷原则相切合，因此，很多项目经理正积极地尝试解决这些差异。当然必须指出，PMBOK 涉及的大部分领域和敏捷是有差异的。**PRINCE2**[②]与 PMBOK 完全相同，PRINCE2 是由英国政府商业办公室发布和维护的一种相似的项目管理方法，主要在欧洲使用。

最后同样重要的是**统一过程**方法[③]，其最著名的改进版本就是 **Rational 统一过程**[④](RUP)，由 Rational 软件(现在属于 IBM)在 1997 年制定。RUP 之于软件开发人员就如同 PMBOK 之于项目经理。RUP 定义了相当全面的过程框架，可以(并应当)剪裁以适应特殊的项目环境，但是从要求交付的文档来看，整个框架看上去还是很官僚，很笨重。敏捷专家认为，过程应当从最简单的少量实践开始，并贯穿整个项目不断地改进。RUP 尝试了相反的方法，一开始就定义了很多实践，然后建议删除那些不需要的实践(我经常把 RUP 方法比喻为为了改装成自行车而去购买波音 747，其实很多项目用自行车就够了)。当然为了应对敏捷在全球的流行，统一过程推出了几种敏捷选择，包括**敏捷统一过程**[⑤](AUP)，**开放统一过程**[⑥](OpenUP)和**精炼统一过程**[⑦](EssUP)。但是它们当中没有一个在全球敏捷联盟中表

① http://www.mgt30.com/pmbok/
② http://www.mgt30.com/prince2/
③ http://www.mgt30.com/up/
④ http://www.mgt30.com/rup/
⑤ http://www.mgt30.com/aup/
⑥ http://www.mgt30.com/openup/
⑦ http://www.mgt30.com/essup/

现良好。

敏捷的障碍

经验证据反复告诉我们，只要做得好，敏捷软件开发会带来巨大的投资回报率[Rico 2009]。但如果敏捷方法有这么积极的作用，为什么还是有人排斥它呢？为什么那么多软件项目还是会失败[①]？

VersionOne 在“2009 年敏捷开发状况调查”报告中指出：“管理层反对变革，管理失控，缺乏工程纪律，团队反对变革，工程人才的素质，加之大多数组织都离不开计划、可预测性和文档，这些都是敏捷应用的主要问题。”[VersionOne 2009]

等等……让我们再次检查一下这些担忧：我们正在讨论各式各样的管理控制，组织变更管理，工程人才……

如果我错了，请原谅，但是……这些难道不全部属于管理职责吗？这难道不正表示管理者才是敏捷软件开发的最大障碍吗？

作为一名管理者，这个结论无法让我心情愉快。

但作为作者，这个结论倒是让我挺开心的。

我认为，敏捷软件开发低估了职能管理的重要性。如果管理者不知道在一个敏捷组织中应该做什么，应该抱有怎样的期望，他们又应该如何参与敏捷软件开发的转型？在这里，敏捷传达了什么信息呢？如果只是“我们不需要管理者”，那么敏捷转型在世界各地都遭到阻扰就不足为奇了。

因此，为了让组织享受到敏捷转型的好处，他们需要知道一个重要问题的答案：在敏捷世界里，管理者路在何方？

① CHAOS 2009 总结报告的出版物可参见 http://www.mgt30.com/chaosreport/

职能经理之于项目经理

我的姓氏在我们国家并不常见。但是不管怎样，在我的职业生涯中，我用过 Jurgen、Jurjen 和 Jörgen。这导致了很多困惑。当名字相似的时候，人们容易忽略其他方面的差异。比如，如果 Ella Fitzgerald(唱歌不错)的名字是 Jurgen 的话，我想我的同事会让我为他们唱歌的。

在那些称为“经理”的人身上，我看到了同样的问题。

2005 年，一群专职管理的人(项目经理、职能经理以及其他经理)聚在一起，拟定了一份**相互依赖声明**(DOI)[①](参见图 2.4)。

最开始的相互依赖声明主要用于项目管理。随后发现这些原则可以诠释得更加宽泛，并且应用于“一般管理”。然而，声明主要面向如何管理软件项目，而不是如何管理团队成员。这一点可以从下面的事实中得到印证：DOI 的作者，也是敏捷项目领导力网络(Agile Project Leadership Network)的创建者[②]。

但是很不幸，项目管理和职能经理的角色通常混淆在一起。即使权威专家出版的优秀书籍也在同时讨论项目管理和职能管理问题，这些书包括《敏捷管理》(Agile Management)[Anderson 2004]、《管理敏捷项目》(Managing Agile Projects)[Augustine 2005]以及《敏捷项目管理》(Agile Project Managment)[Highsmith 2009]。类似的情形也出现在很多论坛、博客和杂志中。我希望两者是不同的，因为项目管理和职能管理不是一回事。但这就如同将软件工程师和系统管理员搞混一样，虽然他们有着相同的思想，讲一样的笑话，留同样的发型，并且穿同样的衣服(更形象的说法)，但是他们不应该被当作同样的人。(我是认真的，你可以试着要求软件工程师帮你修复电脑。但最好不要这样做。)

① http://www.mgt30.com/doi/
② http://www.mgt30.com/apln/

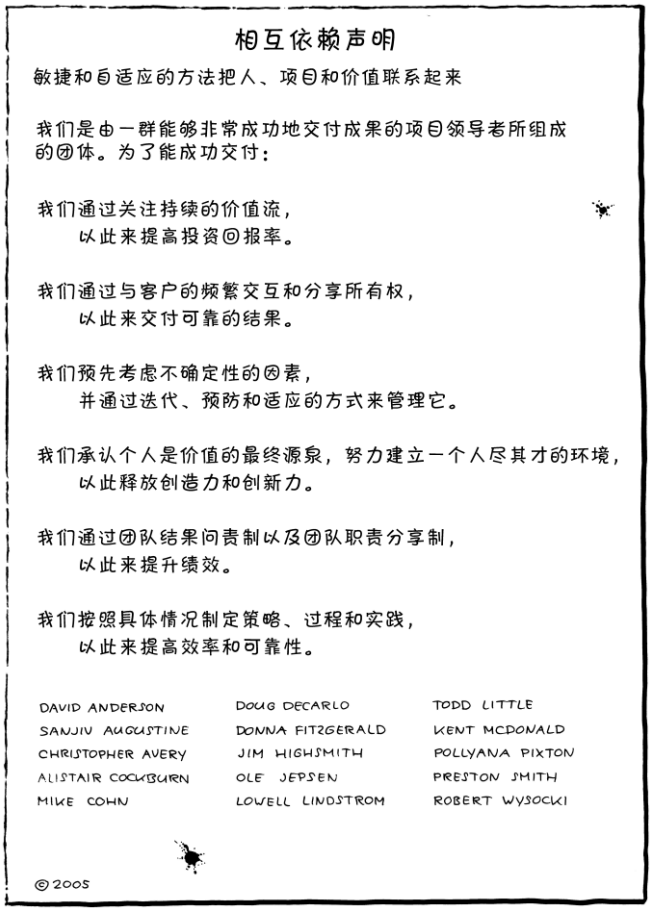


图 2.4　相互依赖声明

因为不能很清晰地区分职能管理和项目管理，所以我们使职能经理和项目经理很难理解在敏捷组织中他们各自的角色是什么。幸运的是，我不是唯一意识到这个问题的人。在我之前，已有出版了几本书这样的书，包括《门后的秘密》[Rothman, Derby 2005]和《领导精益软件开发》[Poppendieck 2009]，它们为软件开发组织中职能经理的职责勾勒出清晰的轮廓。

在本书中，我把职能管理和项目管理两个角色分开。我的主要目标是帮助职能经理(包括开发经理和团队主管)理解他们在组织中的角色。但是我想，本书对项目经理、系统经理、服务经理、行政经理和生活经理也会有帮助。

对于那些认为我是 DJ Jurgen(荷兰著名电台主持人)的读者……真是不好意思了啊！

小结

敏捷软件开发是上世纪九十年代兴起的一种软件开发方法。它是对官僚和随性开发方法的反击，因为那些方法不能持续地成功交付软件产品。

敏捷软件开发，参照敏捷宣言所阐述的价值和原则，始终关注人和团队、持续交付高质量的产品、频繁的客户交互、拥抱变化和最小的前期计划。

敏捷价值和原则体现在不同的敏捷方法中，比如 Scrum 和极限编程。然而，没有一种敏捷方法解决敏捷组织中职能经理(不要和项目管理混淆)的角色问题。这也导致职能经理成为采纳敏捷实践的最大障碍。

反思与行动

接下来，让我们看一看如何将本章的一些观点应用于具体的组织中。

- 检查软件项目的七个维度(人、功能、质量、工具、时间、价值、流程)。你的软件项目是否已通盘考虑这些因素？你的团队在这七个维度上是否都是敏捷的？如果不是，你有什么样的改进计划？
- 想想组织中的管理者。哪些管理者是采纳敏捷软件开发方法的障碍？你能为此做些什么？确定你知道要从他们那里得到什么支持才能保证敏捷管理方法的成功实施。
- 是否每个人都清楚相关人员的职能经理？相比项目经理，是否还有人怀疑或不赞成职能经理这个角色？如果有，应该为此做些什么？
- 通过订阅敏捷团队和组织的博客、讨论组来提升你的敏捷管理技能。也可从本书配套网站 http://www.management30.com 找到最新更新列表。