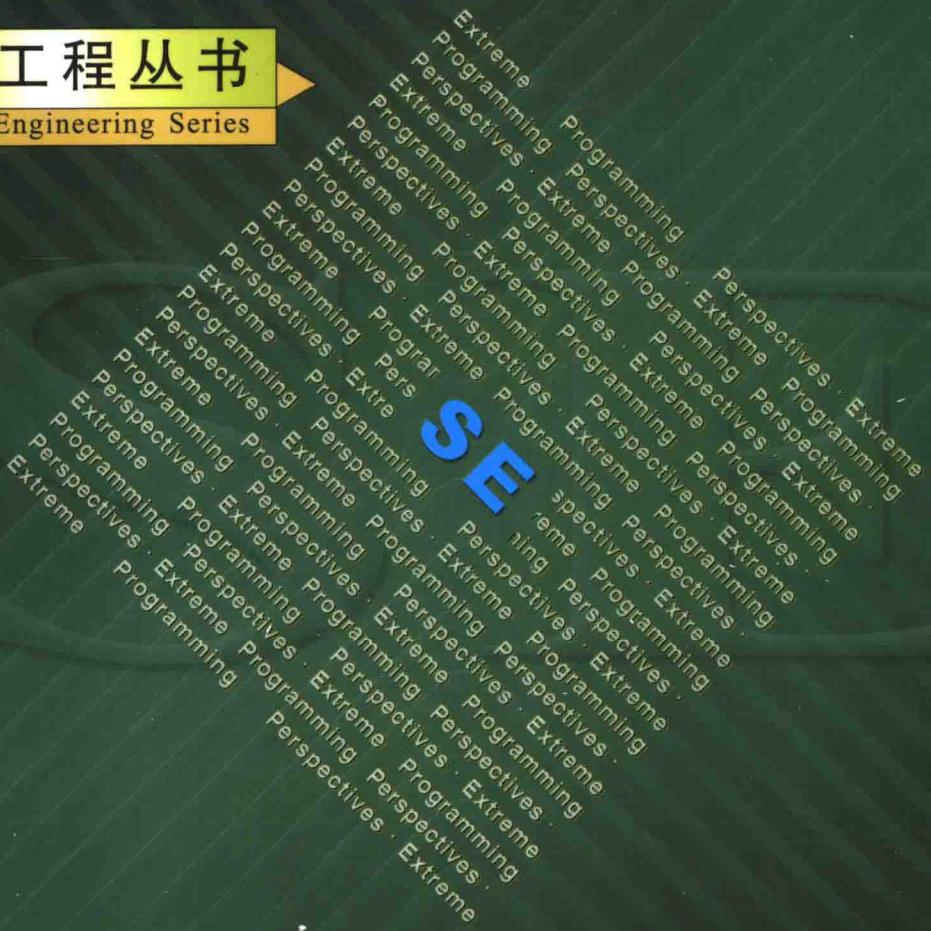
软件工程丛书
Software Engineering Series



极限编程透视

Extreme Programming Perspectives

[美] Michele Marchesi Giancarlo Succi
Don Wells Laurie Williams 等著

卢庆龄 张 威 王小振 等译

PEARSON Addison Wesley
PHEI
電子工業出版社
Publishing House of Electronics Industry
http://www.phei.com.cn

软件工程丛书

极限编程透视

Extreme Programming Perspectives

[美] Michele Marchesi
Giancarlo Succi
Don Wells
Laurie Williams
等著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内容简介

本书收集了47篇有关XP的论文。这些论文共分为六个主要部分：XP和AM概述，XP开发实践，向XP和AM转变过程中所包含的问题，应用XP进行工作的实际经验，如何使用极限工具帮助实际应用XP和AM，最后讨论了一些发展和扩充XP的思想。围绕XP和其他灵活方法论中讨论的多个关键主题，本书提出了有效实施XP的经验性技术，并给出了实现成功转变的策略。

本书收集的论文涉及多个行业中的极限编程实践。对于已经在进行XP开发，或者是准备转换到这种灵活方法论上的人来说，具有较强的指导意义。对于采用传统开发方法的程序员来说，也有助于他们开阔视野，进而接受这些新的编程思想。

版权贸易合同登记号：图字：01-2003-1038

图书在版编目（CIP）数据

极限编程透视/（美）马凯西（Marchesi, M）等著；卢庆龄等译 -北京：电子工业出版社，2004.7
（软件工程丛书）
书名原文：Extreme Programming Perspectives
ISBN 7-121-00061-X

Ⅰ.极… Ⅱ.①马… ②卢… Ⅲ.软件开发-文集 Ⅳ.TP311.52-53

中国版本图书馆CIP数据核字（2004）第062223号

责任编辑：赵红燕
印　　刷：北京兴华印刷厂
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱　邮编：100036
经　　销：各地新华书店
开　　本：787 × 980　1/16　　印张：26　字数：510千字
印　　次：2004年7月第1次印刷
定　　价：42.00元

前　言

为什么我们需要另一本Agile/XP的图书？如果XP和Agile Manifesto都很简洁，那么为什么还需要那么多的书籍、文章、讨论会、用户讨论组、Yahoo分组电子邮件以及辩论会进行阐述呢？这是因为简洁并不表示过于简单，“简洁”的原则和实践的有效组合可以产生复杂的、智能的行为。

XP的12种实践，DSDM的9个原则，Bob Charette的“节俭开发”（Lean Development）中的12条准则，以及与Agile Manifesto有关的12条准则（12似乎是Agilists非常流行的数字）并不简单。这些复杂的问题，受技术和人类能力的限制，不太容易解决，但我们可以通过一些内在的规则、实践和准则很好地逼近，使“应用程序”产生无穷的有创造性的想法，这些想法反过来又可以为我们的顾客带来价值。

这一点很关键，但严格的方法论支持者并不理解。他们中的许多人只相信内在的规则、程序和过程，而不相信生成的规则。如果有问题，可以翻到过程57、活动24、任务87、步骤4，即可找到答案。遗憾的是，复杂的问题并不能通过这些数字来找到答案。复杂的问题，也就是每人每天都要面对的现实问题，即软件产品开发杂乱无章的局面，可以在几个关键原则的指导下，基于几个关键实践，通过有创造性的、革新的想法来解决。正如Kathleen Eisenhardt和Donald Sull在“Harvard Business Review”一文（“Strategy as Simple Rules”，2001.01）中所写的那样，“如果商务前景是简单的，那么公司可以采取复杂的策略，但现在的商务是如此复杂，因此需要对它们进行简化”。

简化并不意味着过于简单，而是意味着要从数以百计的软件开发规则和实践中提取出能够使我们清晰、有效地考虑所面临问题的一些内容。如果实践过于简单，我们将不需要有关单个实践的全部书籍：重构（Martin Fowler），或结对编程（Laurie Williams和Robert Kessler），或测试优先开发（Kent Beck）。

本书各章的差异证明了我的观点。本书由Agile/XP领域的著名领导者和不太为人所知的领导者合作编写，他们每天都在努力工作以为其顾客创造价值。这些章节反映了现实世界问题的复杂性及其解决办法，有助于我们了解几个非常有价值的关键、简单的观点。

类似这样的图书是很有价值的。虽然各个章节不一定适合每位读者，但可以先跳过一些章节，只学习感兴趣的部分，之后再学习其他章节，这样读者可以深

入了解同时代的人如何使用Agile/XP实践来解决众多的现实世界问题。但我们必须承认，参与在地中海撒丁岛海滩举行的XP2001会议的人对本书许多章节的形成都有一定的贡献。

Jim Highsmith

目　录

第一部分　XAR：极限和 Agile 回顾——XP 和 AM

第二部分　XD: 极限开发——XP 开发实践的分析

第三部分　XTT：极限技术转换——XP 和 AM 入门

第四部分　XD：极限真实性——现实生活经验

第五部分　XT：极限工具——工具怎样帮助实践 XP 和 AM

第六部分　XEX：通向极限的极限——关于如何扩展 XP 和 AM 的方法

第一部分

XAR：极限和 Agile 回顾——XP 和 AM

欢迎阅读又一本有关极限编程、当然也是有关灵活方法论（Agile Methodologies，AM）的书籍。

编写一本有关新软件开发方法论的书籍是一项既容易又困难的任务。之所以说它容易，是因为可以用普通和高级朦胧的方式来讨论它，不必涉及细节和采用严谨、正确的语句，例如“它要求定制”以及“没有银弹”等。

如果曾经用自己的方法论开发过项目，那么现在可以用完全不同的方法来开发，而且这种方法也相当容易。采用这种方法，可以详细描述所做的工作，并围绕自己的特性提取出这种完全开发方法的精华。如果添加一些只限于小范围使用的内容，诸如用右手按键、用左手拿餐巾、在工作间隙擦拭羊毛衫等，效果会更好。

软件设计人员天生就具备有利条件。我们不应忘记在电子技术研究的早期，电阻方程是相当复杂的，含有许多无关紧要的条件，而现在，它只是一个简单的代数除法公式：$R = V/I$。

然而，如果目标是为读者建立一些有价值的内容，将有限的理论和有限的经验进行组合，并且还要避免太过简洁的语言，那么编写有关新方法论的书籍是非常困难的。

本书采用了上述方法，将下述内容进行有机组合：XP概述（来自提出XP设想的人们），特定领域的经验描述（有些还不是很清楚，有待进一步讨论），以及软件公司如何控制XP项目进度的经验评价。

第一部分概述了这种方法论。该部分假定读者已经阅读了相关书籍（如Kent Beck、Martin Fowler、Ron Jeffries、Jim Highsmith等人的著作），因此只是在第1章（由Don Wells编写）的概要中介绍了这些基础知识。

在第2章，Jim Highsmith描述了“灵活性”的实质。这一有深刻见解的章节从一定深度探究了灵活方法论，并解释了所有灵活建议背后的基本原理。

太多的灵活方法论可能会使软件管理者和工程师感到无所适从。为解决这一问题，Michele Marchesi在第3章介绍了选择最适合自己环境的灵活方法论的原则。

多数人认为XP可能是与结对编程相关联的，反对者认为这只是一种浪费软件公司时间和金钱的时尚，而支持者则将其视为软件开发的“圣典”。在第4章，Laurie Williams讨论了结对编程的原则，详细叙述了何时及怎样采用结对编程，并概述了预期的益处。

在XP中心，要对查找适当隐喻（metaphor）的能力进行有效分析并不是一件简单的任务，尤其当项目变得更大和更复杂时就更为困难。在第5章，William和Steven Wake集中讨论了隐喻问题。尤其是详细介绍了他们查找隐喻、将系统对象与隐喻相关联以及评价是否有“不好的隐喻”的方法。

XP不是“牛仔编码”，也不是完全失控的开发，而是与之对立的开发方法。但如何将灵活方法与度量和过程控制相结合呢？这似乎是自相矛盾的。在第6章，利用软件工程师和管理者自然使用的数据资源，Giancarlo Succi讨论了实现灵活度量程序的可能方法。

XP的各个实践可用于什么范围？在我们的项目中应该保留多少才是成功的？什么是实践定制的安全级？在第7章，Ron Jeffries试图回答这些管理者每天都要面对的棘手问题。他分析了最相关的实践，并总结了灵活定制的注意事项。

最后，在第8章，Michele Marchesi介绍了如何使用XP达到自己的目标。他解释了如何定位商业和技术目标，概述了实现XP项目可能遇到的问题，并给出了解决这些问题的方法。

第1章　XP概述

—— Don Wells

有人认为XP（Extreme Programming）是一套新规则，也有人认为它是人文价值的集合，还有人认为它是非常危险的过于单纯的事物。XP是在我们发现计算机编程成为瓶颈的时候出现的。商业上需要快速开发较大的系统以取得竞争优势，软件建立的速度通常可以决定新商业机会的增长速度。在过去的几十年中，软件的建立方法已经发生了许多变化，但还没有发生过如此巨大的变化。XP不仅仅是规则和价值，还是软件开发的再设计，以适应行业中新的极限需求。

XP最有争议的基本观点是今天只做今天所要做的事情，即使是温和的读者最初看到这一观点时也可能会跳起来。为打消他们的疑虑，我们需要指出的是，这并不意味着没有设计和计划，这些活动在XP中甚至比在其他方法中更重要。实际上，XP最大的影响在于与时间相关的系统费用。软件行业的一个更重要的方面是生命周期的观点以及与之相关的费用。XP提出了一些新的管理生命周期费用的方法，其中一种方法是对先期交付的资金指定使用期限，在此期限内使每一分钱都发挥最大效能，超过此期限，则停止新的开发需求。

人们对XP的文档问题通常有些误解，认为对任何事情都不需要写文档，这是十分荒谬的。XP不是不写文档——而是写更为有效的文档。在大多数情况下之所以不写文档，是因为可以使用更为有效的形式。小组成员可以记住项目的许多文档，并且可以通过谈话方式与小组的新成员进行交流。文档的修改跟不上小组集体意见的变化，对文档的访问也不如会话那么迅速，但这并不表示缺少可靠的文档。相反，在自动测试中，有大量合适的完好代码和文档。

依靠小组和集体意见的观点确实是有争议的。在不久以前，程序员可以独立编写软件，并且有可能一夜成名。但时代不同了，现在没有人能够独自建立完整的系统，并及时投放市场以创造价值。随着小组的出现，有必要了解小组的工作。XP看重的是小组的协作，整个小组就像一个单独的实体。构成小组的成员可能会发生变化，但小组本身会坚强、稳定、可靠地按预期的目标发展。

如果没有反馈，请不要做任何事情。在日常生活中，我们中的许多人不把这一概念当一回事。我们应该用听到、看到和感觉到的丰富信息来指导我们的行动。软件开发也是如此，只是需要找到适当的反馈源。通过移动鼠标并单击它与程序

进行交互就相当于现实世界的看、听和感觉，但这是远远不够的，我们的感觉不能提供有关待测软件的足够信息。程序并不存在于我们所在的物理世界，因此必须通过编写代码将感觉扩展到软件世界。我们需要用像JUnit这样的工具将软件开发所需的反馈类型转化为我们能够感知的反馈类型。

考虑相关的想法：不要等项目完成之后再进行测试——实际上，测试要首先进行。我们需要用反馈来监视项目的进展和指导项目的开发。知道测试什么和如何测试是一种有价值的技能。一般不要求程序员测试自己的软件，测试工作通常由单独的质量保证（QA）组来承担，甚至让没有嫌疑的客户来查找问题。要快速交付软件，就必须改变这种方式。程序员必须充分了解测试，甚至应有在编码之前就建立测试的观念。这样无论有没有QA，小组都能够完全彻底地负责测试工作。

XP项目的管理遵循计划和反馈的同心循环。“测试第一，编码第二”的循环就是一个这样的例子。测试是下一步要编写什么代码的计划，我们通常会忘记计划是一种预测，而这种预测不是基于历史数据，而是基于猜想。了解某件事情需要花费多长时间的惟一方法是对其进行度量。计划也是建立在度量的基础上的，并且一旦有了新的度量方法，就要对计划进行修订。因此建立正确的度量方法是非常重要的。要充分利用所拥有的数据来预测未来，而不要让猜想蒙蔽了你的双眼。

XP涉及一些新的规则，一些旧的被人遗忘的规则，更多的人道价值，当然还有小组工作，但还有更多未被注意的内容，还有许多如何将工作做得更好、更可靠的新的观念。我们必须停下来想一想我们正在做什么，并且问一下为什么要做这件事情。如果XP中没有这些内容，我们就会失去发现新方法的意识，而这些意识有助于我们开发更多的软件，并且少走弯路。

第2章　灵活软件开发

—— Jim Highsmith

去年，随着对XP的持续狂热，在开发领域涌现出灵活软件开发的热潮。这一开发方法的酝酿已经有10年之久。在过去的几年中，有关灵活开发的文章已经出现在“Computer World”、“Software Development”、“IEEE Computer and Software”、“Cutter IT Journal”、“CIO”等期刊中，甚至还出现在“The Economist”中。笔者认为，灵活开发突然流行的根源在于两件事情：在持续变化和动荡的时期，灵活软件开发有助于公司交付有价值的软件；在信息时代，灵活软件开发有助于建立工作场所文化，以吸引具有丰富知识的工作人员。

在我们的专业中，评价这一热烈争论的趋势，需要把握3个关键问题：灵活性最适于解决什么类型的问题？什么是灵活性？什么是灵活软件开发生态系统？

灵活性最适于解决什么类型的问题

信息时代经济的未来属于“灵活性”。公司要有能力使其竞争对手发生变化，这种变化甚至可能有些混乱。如果能够快速良好地进行革新，就能迫使竞争对手发生变化。如果能够在竞争初期对新技术、客户需求做出快速响应，也会迫使竞争对手发生变化。如果行动缓慢、缺乏创新以及响应迟钝，就会被别人牵着鼻子走。你是想自己设定变化的步骤，还是想让竞争对手设定？在信息时代的经济领域，公司设定步骤、进行变化的能力，取决于开发软件的能力。在持续变化的世界中，传统而严格的软件开发方法是难以获得成功的。

在最近几年，软件技术已经从支持商业运作转变为商业策略的关键组件。它可以驱动新产品的开发，从含有数百个芯片（含内嵌程序）的汽车，到便携式电话和其他无线设备，它们正在扩展“分布式”系统的定义。

为顺应革新并做出快速响应，灵活软件开发（Agile Software Development, ASD）应运而生，它可以建立能为商业带来价值的新知识，并能够快速响应竞争挑战。严格软件方法论（Rigorous Software Methologies, RSM）也是有用的，但应用的范围有所减少。RSM的许多技术可以有效地被ASD方法所采用，但两者的构架和基本原理是不同的。灵活方法最适用于开发新产品和增强小组力量，其中革新和创造力是极为重要的。

人们通常只关注Kent Beck所著“Extreme Programming Explained: Embrace Change”[Beck2000]一书的标题，而忽略了副标题。这些人，尤其是大型组织的管理者，可能会对“极限”一词及其内涵产生误解，认为这是鲁莽者的活动。还可能在跳过“极限”一词而看到“编程”一词时，将其归入令人讨厌的“机械”素材中。但Kent在其著作中用这些词来讨论编程，甚至倡导用这样的极限实践来测试自己的代码。围绕XP的战略问题，以及所有其他灵活软件开发生态系统（ASDE），都涉及变化的问题。

灵活的组织可以为其竞争对手制造混乱。首先，尽快制造变化，让竞争对手疲于奔命；其次，对竞争对手改变市场的企图做出快速响应。想象一下，如果竞争对手开发新产品的周期是12个月，而你要用18个月，那会是什么感觉？还有，每次当你推出具有突破性的功能时，竞争对手就会在其产品的下一个版本中推出与之相应的功能，这样他们就会掌握主动权，而你总是处于被动防守的地位。这就是灵活组织的实质——创造能使你生存而竞争对手不能生存的变化。Silicon Graphics的CEO Ed McCracken说，“企业竞争的源泉就是管理混乱环境的能力，但是要更加主动，我们实际上是首先帮助制造混乱——这是摆脱潜在竞争对手的方法”[Iansiti1998]。灵活的组织不仅要响应变化，而且要产生变化。

什么是灵活性

如果问题是动荡和混乱的，那么灵活性就是解决问题的关键。

灵活性不是核对组织的初始列表的一次性处理，而是一种生活方式，一种响应商业动荡的持续显现和变化。批评者认为，“灵活性仅仅等待坏的事情发生，然后进行响应。它是缺乏计划和风头主义的代名词。”但灵活组织也有计划，只是他们了解计划的局限性。3个特征有助于定义灵活性：制造和响应变化；行动敏捷并能够临时准备；能够在机动性和结构之间进行平衡。

灵活性是制造和响应变化的能力，目的是在动荡的商业环境中得益。

灵活性不仅仅是反应，也是行动。灵活的组织首要的步骤是制造变化，变化可以为竞争对手施加强大的压力。制造变化需要进行革新，也就是要具有创造提供商业价值的新知识的能力。其次，灵活组织还要有响应能力，在商业环境中，要能够快速有效地响应预期和非预期的变化。

在不稳定的商业环境中，公司需要在其组织的各个层次上增强“探险”技能。优秀的探险者就是灵活的探险者——他们知道如何巧妙处理和临时准备。印第安纳州的Jones就是一个优秀的探险者，不知何故他历经多次奇异的冒险仍能够存活下来。灵活性意味着快速、轻巧和敏捷——迅速采取行动的能力，利用最少的资源完成任务的能力，适应变化了的环境的能力。灵活性还要求革新和创造力——

预见商业上新产品和新方法的能力。特别是，IT组织对探险和最优化需求之间的平衡，还没有进行足够的研究。

灵活的个人能够临时准备——他们知道规则和界限，而且还知道手头的问题何时会进入未知领域。他们知道如何通过试验和学习，将其知识扩展到未预见到的领域。当要处理危急情况时，能够召集最有应急能力的人。

即席创作产生了伟大的爵士乐队。通过几个关键的结构性规则，爵士乐队能够演奏出各种即席作品。由于他们拥有坚实的基础，因而具有巨大的灵活性，而且不会使音乐变为噪音。商业过程再设计和软件工程方法论的支持者可能赞同应急能力是成功的关键这一观点，而不是小心关联的过程。在当今动荡的环境中，具备良好的平衡、判断和应急能力的价值确实是无法估量的。

灵活意味着要信任某个人的响应能力，而不要信任其计划能力。

什么是灵活软件开发生态系统

在笔者开始编写有关灵活软件开发方法论的书籍时，始终担心“方法论”一词，因为它并不能准确描述灵活开发的焦点：人员、关系和不确定性。进一步说，如果使用“方法论”一词，人们会立即将灵活实践与传统软件开发方法论比较，从而使用错误的度量方法进行比较。因此笔者开始使用术语“灵活软件开发生态系统”来描述整体环境，其中包括3个交叉的组件——混乱有序（chaordic）的观点，协作的价值和原则，以及不充分的方法论，而术语“灵活主义者”则用于表示ASDE的支持者。

有些人认为“灵活”意味着较少的过程、较少的仪式和简短的文档，但它还有更广阔的外延，这就是使用单词“生态系统”而不是“方法论”的主要原因。虽然较少的过程和较少的形式可以降低开发费用，但并不足以产生灵活性。集中在人员及其交互上并使每个人具有快速决策和自适应其过程的能力，是灵活生态系统的关键。

单词“生态系统”可使人想起生物及其相互作用的情景。在有组织的环境中，生态系统可以看做一个动态、不断变化的环境，其中的人和组织不断地进行活动并相互响应对方的活动。单词“生态系统”可以使我们集中在个体和小组的动态交互上，而不是组织图上的静态线条。

混乱有序的观点

要充分了解ASDE，我们需要了解它的3个组件及其相互关系。首先，灵活主义者认为组织是混乱有序的——每个组织都具有混乱和有序的特性，这就不能通

过线性方法、预先计划和执行实践来管理。将组织看做混乱有序的意味着我们需要了解传统项目管理和开发生命周期实践的可预测性是客户、管理和开发组织之间功能紊乱的根本原因。混乱有序的观点影响我们如何响应变化和如何管理项目小组及组织。

在日复一日的项目工作中，混乱有序的观点会产生两种结果，它们和严格方法论的同步观念是完全不同的。

- 产品目标是可达的，但却是不可预测的。
- 过程可能是一致的，但却是不可重复的。

尽管ASDE包含详细的计划，但在动荡的环境中，计划中的基本设想是不可预测的，至少在项目范围、进度和费用方面是不可预测的。计划是需要测试的假说，而不是需要实现的预测。但是，商业的产品目标是可达的，这在很大程度上是因为灵活的人可以适应环境。他们通过交替使用其他两个方面来“适应”有关联的情景和进度、范围或费用目标。其次，尽管过程可以帮助人们一起工作，但在不稳定的环境中，通过度量和纠错来驱动过程变化的观点——静态过程控制，变成了不切实际的假说。变化是项目开发期间获取的知识的结果，在项目早期知识是不可认识的，要求有响应变化的过程，而不是试图消除变化的过程。

Peter Senge使用术语“智力模型”来表示这种观点——或我们头脑中的一套设想、素材和信仰，所有这些形成了思考的环境[Senge1990]。在组织中，整套智力模型定义了总的文化环境。面向销售的公司和面向工程的公司是不同的，而驱动策略是客户友好的公司和驱动策略是产品革新的公司也是不同的。智力模型包括线性、原因和结果、层次、可预测性和可控操作的公司，与智力模型包括协作网络、紧急情况、权力分散和不可测验收的公司也有很大的不同。前者遵循牛顿学说，而后者遵循混乱有序学说。

协作的价值和原则

定义ASDE的互联网络的第二部分是有关协作的价值和原则的语句。尽管难以用一个单词来描述Agile Manifesto的特性，但“协作”似乎是一个最好的形容词。价值和原则形成了生态系统。没有一套规定的价值和原则，生态系统就是毫无结果的反映实践，而不是人们能在其中交互的环境。

协作文化包括人员、开发小组内部的关系，以及与客户、管理者和公司内外合作伙伴之间的关系。人力、沟通和协作通常被称为“软”科学，但实际上，它们是最难以掌握的。规则和价值有助于定义文化——即人们想在其中工作的环境。

不充分的方法论

ASDE的最后一个组件是方法论。方法论的传统定义包括诸如作用、活动、过程、技术以及工具之类的内容。Alistair Cockburn在定义方法论时将这些组件总结为“大家同意的约定” ——人们按这种方式在项目中一起工作。在“The Social Life of Information”一书中，John Seely Brown和Paul Duguid讨论了过程（如商业过程再设计运动使用的过程）和实践之间的主要区别（[Brown+2000]）。过程可以用手册的形式描述，而实践是现实中发生的事情。过程论者将人放在第二位，而实践论者将人放在第一位。过程论者将焦点集中在明确的（写下来的）知识上，而实践论者将焦点集中在暗含（内部）的知识上。ASDE模型提出以实践为中心来逼近方法论，而不是以过程为中心。

采用不充分的方法论有两个原因：价值和革新。流线型方法论集中在能创造价值的活动上，并无情地抛弃没有价值的活动。编程通常会带来价值，而过程管理通常会增加开销。不充分意味着保留前者而放弃后者。其次，在混乱有序的环境下（不是有序的环境），革新和创造力起主导作用。不充分的方法论是哺育革新的摇篮。

方法论还与有组织的模型相关。灵活方法论含有最少的过程和文档，以及简化的仪式（形式）。灵活方法论被看做是“不充分的”，或“比刚够略少一点”，或“最小的”。然而，这种流线型的方法论并不只是基于减少工作量，而是（而且是更重要的）基于了解混乱有序的世界这一观点。按照这种观点，可在“混乱的边缘”以最佳方式产生意外（革新）的结果，也就是说居于混乱和有序的中间位置。

实践（或技术）是方法论的源泉，无论是结对编程、Scrum会议、客户焦点组还是自动测试，ASDE的实践都由天才和熟练的个体来执行并产生结果。

灵活软件开发的未来是什么

现在不禁要问，灵活软件开发的未来会是什么样的呢？从某种程度上说，商业环境的未来仍然是动荡的，笔者认为严格的文化会面临严峻的挑战。灵活不是由减少过程或修剪文档产生的——灵活是一种态度，一种对世界如何在复杂形势下运转的感觉。然而，对执行者和管理者而言，他们仍然希望世界是可预知的和可计划的，认为灵活文化和ASDE是难以实现的。但是，通过最终的分析，商业不可避免地会逐渐倾向于使之成功的实践，而且ASDE对成功软件项目的贡献会日益增大。严格的方法还会存在，但从现在起5年以后就很少会有公司使用它们了。