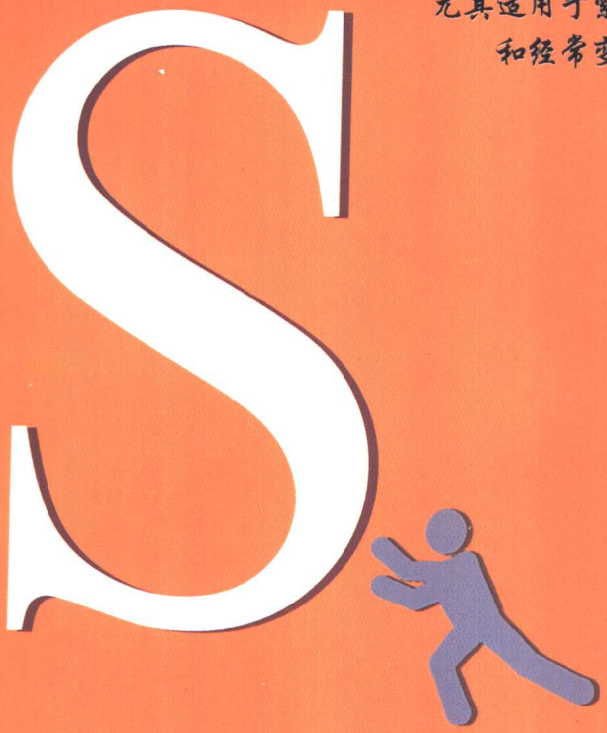
软件管理与软件工程译丛

荣获软件开发杂志JOLT WINNER大奖，
尤其适用于紧张、高度竞争
和经常变更的软件项目



Adaptive Software Development

自适应软件开发

[美]詹姆斯·海斯 (James A. Highsmith III) /著
钱岭 等 /译

清华大学出版社

软件管理与软件工程译丛

Adaptive Software Development

自适应软件开发

[美]詹姆斯·海斯 (James A. Highsmith III) /著

钱岭 等 /译

清华大学出版社

(京)新登字 158 号

EISBN: 0-932633-40-4

图书在版编目(CIP)数据

自适应软件开发: 一种管理复杂系统的协作模式/(美)海斯著; 钱岭等译. —北京: 清华大学出版社, 2002
(软件管理与软件工程译丛)
书名原文: Adaptive Software Development: a collaborative approach to managing complex systems
ISBN 7-302-06053-3

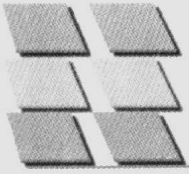
Ⅰ. 自… Ⅱ. ①海…②钱… Ⅲ. 自适应控制—应用—软件开发—项目管理
Ⅳ. TP311.52

中国版本图书馆 CIP 数据核字(2002)第 086586 号

出 版 者: 清华大学出版社(北京清华大学学研大厦, 邮编 100084)
http://www.tup.com.cn
责任编辑: 熊妍妍
封面设计: 孙剑波
版式设计: 肖 米
印 刷 者: 清华大学印刷厂
发 行 者: 新华书店总店北京发行所
开 本: 880×1230 1/32 **印张**: 16.625 **插页**: 2 **字数**: 290 千字
版 次: 2003 年 1 月第 1 版 2003 年 1 月第 1 次印刷
书 号: ISBN 7-302-06053-3/TP·3609
印 数: 0001～8000
定 价: 39.80 元

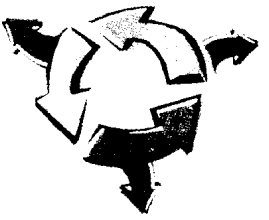
目　录

第 Ⅲ 篇



第 I 篇

第1章



软件增长

杰斐逊山位于俄勒岗瀑布群中，在它高耸的北部山脊，靠近25英尺高冰墙顶部的地方——我左脚穿着登山鞋，鞋钉嵌在冰层中，冰斧插在头顶的冰中有1/4英寸深，右脚在冰岩上摸索着寻找落脚点，身体的后半部分完全悬在离杰斐逊冰河800英尺高的地方——我问自己，“是不是选错山了?”

有各种各样的攀登环境，从午后消遣的出游到挑战人类极限的攀登无所不有。软件项目也一样——从任何一支有经验的队伍就能完成的项目到只有少数世界级的组织才敢去尝试的项目。本书描述了一种特殊的登山情况，并集中考虑了某种特殊形式的软件开发。它处理那些特定领域中的项目，这些项目要求快速而多变，能够突破既

有成果的限制。

在碰到电子商务等典型的快速多变型项目时，各个机构都会陷入困境。和登山者不小心选择了超出他们攀登能力范围的情况类似，许多机构不知道如何处理这些项目，更糟糕的是，他们并没有意识到这一点。他们认为解决方法不过是比以前多做一些工作而已。

即便公司按照常规方法完成了这类项目，其项目小组也会陷入精疲力竭、神情恍惚的状态。如同攀登 K2[①] 或珠穆朗玛峰的登山者，他们步履蹒跚地从峰顶返回大本营，怀着喜悦的心情，充满征服感，只是他们已经耗尽了体力和精神。他们决不想再看其他的山！如同登山运动或独立攀岩运动之类的极限运动，这些项目也要冒很大的风险，任何一个小的失误就可能丧命。本书以极限软件项目为目标，项目中允许犯错误的余地是极其有限的。它不仅打算增加全体职员的生存机会，而且希望保持相对健康的精神状态！

需要用三种同样重要的方法来评估极限项目是否成功。首先，产品能够正常交付；第二，产品达到其预先设定的任务要求，即范围、进度、资源和缺陷级别按某种优先次序的组合；第三，在项目结束的时候，项目小组成员身心还是健康的。

极限项目和那些受速度和不确定性影响较小的项目有着很大的不同。它们需要的不仅仅是一个新的工具或

① 译注：K2 是世界第二高峰。海拔 8 611 米。由于山峰陡峭，是世界上公认的最难攀登的山峰。

技术，也不仅仅是修改过的软件生命周期，更不仅是合作参与的管理。对他们来说，成功有着惟一的含义：

> 成功源自我们对软件的开发和管理，甚至是科学基础的最基本假设的挑战。

为了在这个定义的上下文中理解“成功”的含义，让我们对软件开发的历史做一个回顾。

历史回顾

在创世之初，并不存在所谓方法，此时世界是完美的。随后，产生了宗教信仰，伟大领袖被大众涂上了一层神圣的色彩——DeMarco，Codd，Cox，Yourdon，Orr，Parnas，Booch，这个名单还将继续增加。如此，由领袖领导的方法的年代开始了，世界从“本该如此（Adhocracy）”的状态进入了官僚作风的状态（见图 1-1），也就是从各自打理私人的事情，转变成为每个人都做同一件事情。不过，这决不是领袖们的本意。诚然，许多项目从这些方法中获益颇多，可是更多的项目还在这一片表格、图表和储存库条目的海洋中挣扎。如我的同事 Lynne Nix 所描述的：

> 一个本该如此的方案给了分析员和程序员不去思考的借口，而一个官僚主义的方案则提供给经理们不去思考的借口。
>
> ——L.Nix，Private communication，1996

官僚式软件开发的年代真是不错，那是个由主机系统统治、客户没有过多要求、COBOL 语言为王和信息工程当

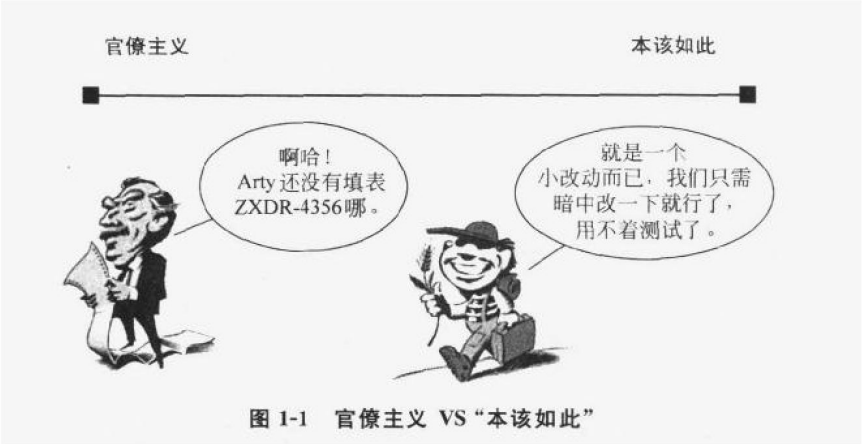


图 1-1　官僚主义 VS“本该如此”

道的年代。我的同事 Ken Orr 将这个时代定义为纪念碑式(monumental)软件开发年代。而在最近的这些年里，使用强有力的个人电脑、C++ 和 Visual Basic 的叛逆的年轻人开创了全新的开发风格——被 Orr 称为即兴式(accidental)软件开发。他们是完全不同的风格，这种对比呈现出当今软件开发社会的分裂状态。

纪念碑式软件开发

纪念碑式软件开发由 20 世纪 70 年代晚期的结构化革命脱胎而来，在这场革命中，模型由数据流图和实体关系图来进行描述。纪念碑式软件开发将以下的要求浓缩为一句话，“任何值得一做的事情都值得做过头”。即

- 由上到下的(top-down)、长期的(long-term)：从组织结构的顶部开始，将商业需求转化为具体的模

型，尤其是数据模型。在数据库中实现这些数据模型，然后构造应用程序。用一年或者更长的时间来进行计划工作，用几年的时间来构造数据库，然后再用几年或者更长的时间来构造应用程序。

- 分析的（工程的）：艰苦而正确地工作以获得正确的软件——一次成型。

这种方案在20世纪80年代的纪念碑式方法论中达到最高峰，它在14卷的细节化任务、文档和表格中定义了开发过程的每一个方面。而这一实践则导致了银弹解决方案（silver-bullet solution）的出现，这一方案称为计算机辅助软件工程（computer-aided software engineering，CASE）。在20世纪90年代，纪念碑式开发的衣钵由软件工程研究所（Software Engineering Institute，SEI）继承，并赋予它一个新的名字——过程改进。

纪念碑式技术曾开发出许多可工作、有效的软件产品，这些产品同样也继承了此技术所带来的一系列缺陷：

- 客户不满意，并且有的勃然大怒。在如此长时间的开发周期末尾，有大量的应用不满足客户的需求；在程序开发过程中，许多需求就已经过时了。
- 技术实现花费太长的时间。商业变化的步伐产生的加速度对于纪念碑式发布周期来说实在过于迅速了。一个服装业的客户花费18个月的时间来进行新系统的准备工作，而当他完成产品需求的时

候，却发现所要构造的系统早已过时了，而打算参与其中的大部分客户已投入到其他的工作中了。

- 技术违背了行政现状。这些宏伟的计划经常会从内部自行分裂。起初签署协议担任负责人的执行者随着时间的推移却变成了批评家。纪念碑式的项目对行政中心的依靠过于强烈，而软件开发组织的存在则往往十分短暂。

简而言之，纪念碑式的开发方法没有很好地适应快速变化的条件。

即兴式软件开发

从另一方面来说，即兴式软件开发被刻画为随意（Ad hoc）或者说没有方法，并且坚信过程只会减缓进度。这种方法是：

- 自下向上的（bottom-up）：一开始就针对客户群的现有需求，立即实现满足这些需求的应用程序，很少考虑全局性问题以及与其他产品的整合问题。
- 短期的（short-term）：将开发周期限制在 2～6 个月的时间框架中。这里通常存在一个假设，即由于商业变化是如此之快，任何开发周期超过 6 个月的项目只会得到一个过时的产品。很多这样的应用系统从一开始就注定要被废弃。
- 问题驱动的（issue-driven）：将应用程序的正确与否