HZ BOOKS
软 件 工 程 技 术 丛 书
前沿论题系列

敏捷建模
极限编程和统一过程的有效实践

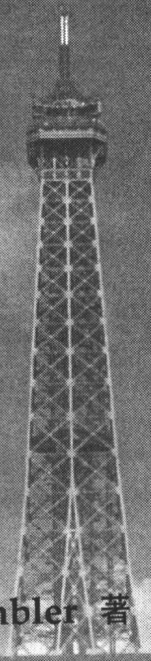
Agile Modeling
Effective Practices for eXtreme Programming and the Unified Process

Scott W. Ambler 著　　张嘉路　等译

机械工业出版社 China Machine Press
中信出版社 CITIC PUBLISHING HOUSE

软 件 工 程 技 术 丛 书

前沿论题系列



Scott W. Ambler 著　张嘉路　等译

敏捷建模

极限编程和统一过程的有效实践

Agile Modeling

Effective Practices for eXtreme Programming and the Unified Process

机械工业出版社 China Machine Press

中信出版社 CITIC PUBLISHING HOUSE

敏捷建模（AM）是一种基于实践的过程，它描述了怎样才能够成为一个高效的建模人员。本书研究了AM的价值观、原则和实践，描述了用来提高建模人员工作效率的技术，而且书中还重新思考了与软件开发有关的几个重要问题，例如，怎样编写文档、怎样组织建模会议和建模团队以及UML适用于什么地方等。此外，还详细研究了怎样在XP项目中有效地建模，并解释了怎样在采用Rational统一过程（RUP）或者企业统一过程（EUP）的项目中简化建模工作。本书既适用于想知道在XP项目中怎样建模以及在RUP项目中怎样简化建模工作的开发人员和建模人员，也适用于想了解“敏捷开发”的项目经理和过程专家。

Scott W. Ambler: Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process（ISBN: 0-471-20282-7）.

本书版权登记号：图字：01-2002-2036

图书在版编目（CIP）数据

敏捷建模：极限编程和统一过程的有效实践 / 阿姆布勒（Ambler, S. W.）著；张嘉路等译. -北京：机械工业出版社，2003.4
（软件工程技术丛书-前沿论题系列）
书名原文：Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process
ISBN 7-111-11700-X

Ⅰ.敏… Ⅱ.①阿… ②张… Ⅲ.软件开发-建立模型 Ⅳ.TP311.52

中国版本图书馆CIP数据核字（2003）第010549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：王高翔 迟振春
北京昌平奔腾印刷厂印刷·新华书店北京发行所发行
2003年4月第1版第1次印刷
787mm×1092mm 1/16·20.25印张
印数：0 001 - 5 000册
定价：45.00元

译者序

随着对软件的需求越来越大，要求越来越高，各软件开发机构也越来越迫切地需要能够更有效地开发更高质量软件的方法。在国内因为开发大型软件的经验相对较少，这一点尤其显得突出。

在这种情况下，有助于提高软件开发有效性和质量的工具引起了普遍的关注，特别是软件开发过程、模型和文档。有的观点认为，只要有了好的过程，开发软件就可以像传统的工厂制造产品那样简单。但并非每个人都同意这个观点。特别是如何建立和贯彻适合于自己组织和项目的软件开发过程，更是仁者见仁智者见智。由于工作的关系，我们接触了不少产品/项目经理、投资人以及一些开发和测试人员。以下是我们常常听到的议论："这个开发过程看起来不错，但我们企业太小，没有人手来实行它"，"我也不知道为什么要写这些文档，反正经理要求写，写就是了"，"客户总是变来变去，项目简直没法进行"或者"我们的项目一天到晚开会讨论，我都没时间写代码了"，问题究竟出在哪里呢？

相信您在读完本书的第1章之后，一定会找到答案或者说有所共鸣，并且迫不及待地想继续寻找解决之道。在本书中，作者Scott Ambler基于自己丰富的经验和敏锐的洞察力，就如何在软件开发过程中激发人的创造力和主观能动性并加以管理给出了一些原则和实践。不仅如此，他还结合当今软件开发界最常用的Rational统一过程（RUP）和极限编程（XP）举出了大量生动有趣的实例。那么，您真能够在其中找到制胜法宝吗？其实，我们翻译本书的目的就是希望能够帮助读者对软件开发中的过程、建模、文档等问题有一个更全面、更客观的理解。至于您是否找到了一颗适合您的"银弹"，那就全在于您对敏捷建模的体会了。

本书由张嘉路、朱鹏、程宾共同翻译，在翻译过程中得到了尤晋元教授的大力支持和悉心指导，张宏、曾玲玲、王跃华等也参与了本书的译校工作，在此深表感谢！

由于时间和水平有限，难免会有一些错误，希望广大读者予以指正。

译 者

2002年12月

序

在Scott请我为这本书写序时，我感到既高兴又惊讶。我感到高兴有几个原因：能够被别人重视总是一件好事情，能够与一本写得这样好的书联系起来当然更好，并且敏捷方法（特别是极限编程）目前也是我自己关心的重点。我感到惊讶，原因在于我是最早和最大声地向Scott“建议”不要把精力放在这个问题上的人之一。“我会进行解释，可说来话长，还是让我概括一下吧。”

对我来说，软件开发最好是在专业化程度尽可能低的情况下完成。我提倡并且相信，最好的结果来自于这样的人组成的团队：他们尽自己所能在任何地方以任何方式做出贡献，而不考虑谁是架构师、建模人员、设计人员、程序人员、测试人员。这不是说我们都必须成为某种像神一样多才多艺的软件人才，而是说我们应该一起同心协力尽自己所能在任何方面做出贡献。

而且，软件开发（和前面一样，对我来说）最好在尽可能少建模的情况下完成。软件开发的意义在于开发软件，在准备工作上花的时间过长，留给重要部分——实际创建坚实的、设计良好的、高质量的软件的时间就会过少。软件的质量与在开始建造系统之前用最新建模工具画的模型并非总是相关的。

由于上面的原因，在Scott提议并启动他的敏捷建模论坛和网站的时候，我是反对这个主意的。我担心注意力过于集中在建模上会减少对整合团队和构建优秀软件所需技能的关注。我在Scott的论坛和私人电子邮件中就是这么说的，一点都不含糊。

事情的结果是，Scott认识到了一些我没有认识到的东西。如果人们要一起合作用敏捷的方式开发软件，大家仅仅有爱、理解和激情是不够的，还必须有技能。团队必须要有分析、建模、设计、编程、测试等所有作为优秀软件基础的各个方面的技能，他们在使用这些技能的时候必须始终与敏捷软件开发的价值观保持一致，这些价值观中的一条就是我们在敏捷宣言中所写的：“个人和交互高于过程和工具。”

这就是Scott在他的论坛、网站和这本书中开始做的事情。他向我们解释如何在敏捷项目的环境中进行有效、轻量级的建模；向我们解释如何在非敏捷的项目中进行建模，着眼于使非敏捷的项目变得更加敏捷。最重要的是，他做这些事情是基于一整套价值观、原则和实践的，而这些价值观、原则和实践能够贯穿到个人或团队的建模方法之中。在关注你所需要的详细的建模技能和实践的同时，他始终注视着软件开发竞争的全局。

Scott讨论了简单工具的使用、所需工作空间的类型、团队所需的组成方式，以及它们怎样结合成一个整体。我特别喜欢引用电影《Starship Troopers》（星舰战队）中Rasczak中士的那句话：“我只有一条纪律：每个人都战斗，不许有人退出，否则我会亲自毙了他。”Scott把建模与从极限编程到统一过程的各种事情都联系了起来，并且做得非常好。

写序的职责之一就是指出谁应该读这本书，我的建议是：如果你是一个需要将建模技能作为开发工作一部分的软件开发人员，也就是说，如果你是软件开发人员，那么就应该读这本

书；如果你是一个需要让自己的工作成果适用于当前迅速变化的软件开发的建模人员，也就是说，如果你是建模人员，那么就应该读这本书；如果你是一个需要了解敏捷软件开发对你的项目意味着什么的软件开发经理，也就是说，如果你是软件开发经理，那么也应该读这本书。无论你在以何种形式参与当今的软件开发，这本书都能够对你有所帮助。

Scott的这本书讲述了在需要迅速地向项目关系人交付高质量软件的软件项目中，进行有效地建模所需的技能。干得不错，Scott!

Ron Jeffries
XP运动先驱

前　言

如果你正在读这个前言，那你很可能是想确定是不是应该买这本书，我喜欢你的态度！为了帮助你做出这个决定，我将很快地回答几个你最可能会问的重要问题。

什么是敏捷建模

敏捷建模（Agile Modeling, AM）是一种基于实践的过程，它描述了怎样才能够成为一个高效的建模人员。当前的建模方法经常被证明有“功能障碍”，一个极端是根本就没有建模存在，当软件被证明是没有经过很好思考的时候，这经常会导致大量的返工；另一个极端是生成过多的模型和文档，这让开发工作慢得像蜗牛一样。AM帮你找到建模的最佳点，在这一点上你既进行了足够的建模，以保证有效地研究和记录系统，但又没有过多地建模以致变成减慢项目进度的负担。

想要在软件开发上采用敏捷方法的项目团队能够并且也应该使用AM的技术，特别是那些遵循极限编程（XP）、DSDM、SCRUM或FDD等敏捷过程的团队。即使在没有采用纯敏捷方法的项目中，也能用AM来改进而且能够经常简化建模工作。

这本书讲了什么

这本书一开始研究了AM的价值观、原则和实践，描述了用来提高建模人员工作效率的技术。尽管可能害怕向别人承认，但你很可能会发现，自己实际上已经在遵循其中的一部分实践，你也很可能会发现新的有效建模的方法。这本书也重新思考了与软件开发有关的几个重要问题，例如，怎样写文档、怎样组织建模会议和建模团队，以及UML适用于什么地方等。正如这本书的标题所表达的，它详细研究了怎样在XP项目中有效地建模，与你可能听到过的恰好相反，建模是XP的一个重要组成部分。这本书还解释了怎样在采用Rational统一过程（RUP）或者企业统一过程（RUP）的项目中简化建模工作。

这本书没讲什么

这本书没有告诉你怎样创建模型。例如，它没有描述写用户故事、用例以及业务规则的步骤。这本书也没有打算成为对UML、数据建模或者以用法为中心的设计（usage-centered design）的介绍。这本书着眼于更大的图景，着眼于建模的过程，而不是微小的细节。这非常与XP描述了开发软件的过程，但没有描述怎样实际去写程序非常类似。

而且，这本书与你以前读过的任何关于软件建模的书都不同。以前讲建模方法学的书一般是先描述几个建模制品，例如，用例、顺序图、类图等，然后描述一个使用这些制品建模软件的方法。AM采取了一种完全不同的方式，它描述了建模的技术，但对于创建模型的类型并不坚持要求。AM只是建议你学习怎样使用种类广泛的建模制品，并努力随着时间不断往你的知识工具箱里加入更多的东西。在其他建模技术随着其下层技术的变化而消失的同时，我相信你会发现AM的原则和实践能够经受住时间的考验，因为它们是真正基础的东西。

我是谁

虽然住在加拿大多伦多北面，但我却是位于丹佛市的Ronin International公司的高级顾问和总裁。我从20世纪80年代中期开始就一直在开发软件，从20世纪90年代初开始就一直在建造面向对象软件。我积极地与客户合作创建用于关键任务的软件，并且利用业余时间在书、杂志和在线白皮书中写一些与自己经验有关的内容。最近几年我一直致力于软件过程问题，包括统一过程（UP）等指令性过程以及极限编程（XP）等敏捷过程，帮助各种组织机构，使他们在软件开发方法上变得更加有效。我也喜欢在软件项目中扮演积极的角色，身为高级开发人员或团队负责人，只要可能我就会卷起袖子干活。

你是谁

你很可能是一个开发人员或建模人员，很想要提高自己作为一个软件专业人员的效率。你想知道在XP项目中怎样建模，或者在RUP项目中怎样简化建模工作。你甚至有可能是一个项目经理或者过程专家，想要了解“敏捷开发这东西”到底是怎么回事。

目　录

第三部分 敏捷建模和极限编程（XP）

第四部分 敏捷建模和统一过程

第五部分 展望

第一部分

敏捷建模简介

这一部分通过对敏捷建模（AM）的价值观、原则和实践的详细讨论，奠定了本书的基础。本部分包含下列章节：

- **第1章　绪论**。这一章略述了软件开发人员当前所面临的挑战，以及敏捷软件开发和敏捷建模是如何解决这些问题的。
- **第2章　敏捷建模的价值观**。这一章讨论了敏捷建模的五项价值观。
- **第3章　核心原则**。这一章详细描述了敏捷建模的核心原则，这对敏捷建模人员而言是非常重要的。
- **第4章　补充原则**。在这一章中，我们将讨论用来支持和增强敏捷建模核心原则的补充原则。
- **第5章　核心实践**。在这一章展现了敏捷建模的重要实践，你必须全盘采用它们，这样才能宣称自己是在进行敏捷建模。这些实践描述了如何采用增量和迭代的方式建模，如何支持和增强团队协作，如何使事情尽可能地简单，以及如何用敏捷的方式验证你的工作。
- **第6章　补充实践**。这些实践描述了创建一个敏捷模型的动机，如何保持文档记录工作的敏捷性以及如何提高敏捷建模人员的生产率。
- **第7章　从混乱到有序：AM的实践如何结合到一起**。这一章概述敏捷建模实践如何结合在一起成为一个协同作用的整体。

第1章
绪　论

要改变命运，必须首先改变自己的态度。

当前，软件开发的情况并不理想。很多系统最终不能交付，或者最终交付的系统经常性地发生延期或超出预算；系统常常不能满足用户的需要，其结果是不得不一遍又一遍地开发。这些问题使客户感到愤怒；因为他们过去被伤害的次数已经太多，他们既不愿意信任我们，也不愿意与我们一起工作。使情况变得更糟的是，客户对我们做什么、怎样做、为什么做没有很好的理解，最终结果是他们把不切实际的要求强加给我们，又不给我们实现他们的目标所需要的支持。

开发工作对软件开发人员来说也不是很有趣。需要工作很长的时间，经常每周50、60、70小时，结果是很快累倒了。经常是在项目开始以前，一旦遇到麻烦，我们就伸出手指指责别人。那些容易被指责的目标包括："愚蠢粗暴的老板们"，我们认为他们的能力勉强够给自己系鞋带；在办公室另外一边的部门里要求过多文档的"纸堆里的呆子"；"愚蠢的用户"，他们经常不知道自己想要什么，在他们确实告诉我们他们想要什么的时候，这些要求又总是没有意义的。当然，我们从来不指责自己，毕竟我们是完美的嘛。那么在意识到项目遇到麻烦时，我们一般做什么呢？有时付出自己的所有，加班加点，为满足交给我们的不切实际的要求而做徒劳的努力，从而踏上一条死亡之旅（death march）（Yourdon 1997）。有时与项目完全脱离关系，做一些不相干的事。因为知道项目注定要失败，所以我们决定至少要学些东西来充实自己的简历，于是从网上下载一些新的开发工具并开始试用。

我们为什么会把自己弄到这样一个悲哀的境地呢？首先，很多人忘记了这样一个事实：软件开发的首要目标是用可能的最有效和效率最高的方式，建造满足用户需要的系统。为了讨论方便，这里所说的系统包含了软件、文档、硬件、中间件、安装过程和操作过程。同样地，很多组织机构忽略了提交软件给客户的紧密衔接的过程，他们把IT（信息技术）部门组织成一些具有专家角色的小组或团队，这是很不幸的，因为这样的小组常常缺乏全局眼光。我怀疑发生这些情况的原因是整整一代的（如果不是两代的话），信息技术专家都相信：为了开发软件，他们必须按照一系列事先定义好的活动来执行。可以根据现在所知道的把这些活动称为指令性（prescriptive）软件过程，也称为重量级（heavy weight）软件过程。这些指令性软件过程同统一过程（Unified Process）（Kruchten 2000; Ambler 2001b）、OPEN过程（OPEN Process）（Graham, Henderson-Sellers, and Younessi 1997）、面向对象的软件过程（Object-Oriented Software Process）（Ambler 1998; Ambler 1999）一样都有它们的位置，问题只是它们可能没有其支持者们所认为的那样正确。这些方法的问题在于，它们通常以指令性过程和应该创建的制品为中心，这是那些把人看作是"即插即用"的组织机构经常实施的做法。换句话说，他们的信仰是：只要有恰当的过程和必要数目的制品，就可以比较容易地把人换进换出项目。我的经

验是：只有在被替换的人工作效率不是很高时，前面的论点才是正确的，这恰是遵循重量级过程的组织机构中常见的情况。现实是：不管遵循哪个过程，替换一个工作效率高的人都是困难的，因此“即插即用”在最好的情况下也是值得怀疑的。

关于指令性过程有趣的一点是，它们通常对管理层很有吸引力，而对大部分开发人员却没有。指令性过程通常基于一种命令和控制（command and control）的模式，它使管理层对事情保持控制，或者说，至少让他们觉得他们在对事情保持控制。指令性过程也倾向于让管理层认为他们能够把项目关系人在软件开发中的作用减到最小，请项目关系人进来参加很少几个短的需求会议，然后忽略他们，直到需要他们进行用户验收测试为止。另一个问题是当形势变得严峻时，开发人员很快就彻底抛弃了过程，不幸的是在他们这样做的时候，好的和坏的一起都被扔掉了，他们常常会发现自己处于一个比以前更大的混乱之中。我的经验是：捷径常常会把你引到泥潭，使你的死亡之旅变得更糟，而不是使你得救。

我也认为开发人员们学习他们职业技能的方式有几个独特的“功能障碍”。在很大程度上，在针对入门水平培训开发人员这件事上，大学和学院做得是比较好的。然而，即使学校的工作是完美的，每个人都得到了学位或文凭，我还是怀疑由于软件开发人员内在的天性，还会有一个问题。在软件开发人员还很年轻（十几岁二十出头）的时候，他们通常集中精力学习和使用技术，称自己为PERL程序员、Linux专家、Enterprise JavaBeans（EJB）开发人员、.NET开发人员等。对他们来说技术是最重要的事情。因为技术在不断地变化，年轻的程序员倾向于大致学习一个技术，在一到两个项目中使用，然后重新开始学习新技术或者以前用到过的技术的最新发展。这里的问题是，他们一遍又一遍地重复学习的不过是同样的低层次基本技能的不同风味。

幸运的是，很多开发人员在经过了几轮技术学习之后逐渐意识到：一旦用COBOL、Java、C#等语言为事务控制编写过代码，就会开始认识到基础的、本质的东西是不变的。不同环境下的数据库访问、用户界面设计等领域也是同样的情况。不久以后，开发人员逐渐认识到无论具体的技术怎样，很多基础性的东西是保持不变的，这些基础性的东西有的在学校里讲过，有的没有。这种认识经常发生在开发人员接近三十岁或刚过三十岁的时候，通常是人们开始稳定下来，结婚、买房的时间。这是比较幸运的情况，因为上面提到的这些新的个人需求意味着他们不可能再投入大量的时间去学新技术，他们需要用这些时间和家庭成员在一起。突然地，高层次的角色如项目负责人、项目经理、（非敏捷）建模人员等对他们变得非常有吸引力，因为这些角色不需要花费持续的大量的精力去学习新技术。于是，等到开发人员开始真正学到技艺的时候，他们已经处于离开开发人员角色的转变过程中了。所幸的是，新的“小年轻”不断跟上来，这个过程在不断地循环重复。最终结果是：大部分最活跃的正在开发软件的人通常不是最称职的做这件事的人，而他们自己甚至还不知道。

业务这一边的情况也同样不妙。客户不理解软件是怎样开发的，停下来想一想，这其实是很合理的。我的经验是，仅仅因为软件开发惊人得困难，造成很少有软件开发人员能够说出软件从始至终是怎样开发的，并能对这个过程中会遇到的不同选择所隐含的结果表现出适度的理解。而且，客户一般对参与他们没有很好理解的复杂过程并不真正感兴趣，在这样的情况下，他们宁愿把细节留给我们以便他们能回去做自己的事情。他们的参与仅仅局限于在项目开始时