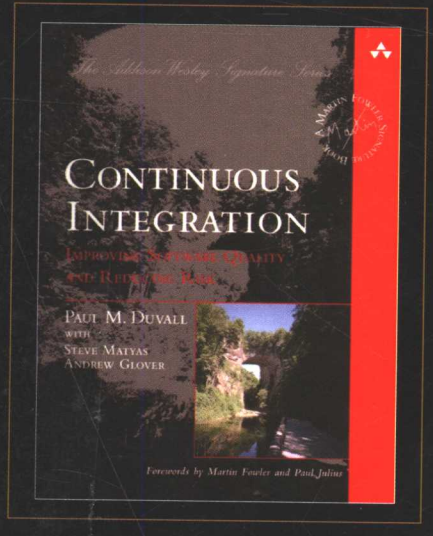
HZ BOOKS
PEARSON Addison Wesley
华章程序员书库

持续集成

软件质量改进和风险降低之道

Continuous Integration

Improving Software Quality and Reducing Risk



（美） Paul M. Duvall Steve Matyas Andrew Glover 著
王海鹏 贾立群 等译

对于希望执行这些最佳实践的人来说，这是一本很好的参考手册。

—— Martin Fowler

机械工业出版社
China Machine Press

华章程序员书库

持续集成

软件质量改进和风险降低之道

Continuous Integration

Improving Software Quality and Reducing Risk

（美） Paul M. Duvall Steve Matyas Andrew Glover 著
王海鹏 贾立群 等译

机械工业出版社
China Machine Press

本书全面深入地讨论持续集成的各个方面。本书介绍了一种增加项目可见性、降低项目失败风险的有效实践。许多软件开发的资深人士认定，这种方法非常不错。本书除了介绍持续集成的基本原则和工具之外，也介绍了测试驱动、代码审查、数据库集成、信息反馈等实践和工具。书中的各种主题介绍了今天在持续集成领域中运用的各种方法，帮助读者衡量需要进行的折衷。

本书适合软件开发人员及团队阅读，也适合大专院校相关专业师生参考。

本书版权登记号：图字：01-2007-4204

图书在版编目（CIP）数据

持续集成：软件质量改进和风险降低之道/（美）杜瓦尔（Duvall, P. M. ），（美）迈耶斯（Matyas, S. ）（美）格洛弗（Glover, A. ）著；王海鹏，贾立群等译. -北京：机械工业出版社，2008.1

书名原文：Continuous Integration: Improving Software Quality and Reducing Risk

ISBN 978-7-111-22921-6

Ⅰ. 持… Ⅱ. ① 杜… ② 迈… ③ 格… ④ 王… ⑤ 贾… Ⅲ. 软件质量-质量管理 Ⅳ. TP311.5

中国版本图书馆CIP数据核字（2007）第185354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：李东震

北京牛山世兴印刷厂印刷 · 新华书店北京发行所发行

2008年1月第 1 版第 1 次印刷

186mm × 240mm · 15.25印张

定价：35.00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（010）68326294

译 者 序

软件项目开发有两大难题：一是确定软件的需求，即确定目标；二是确定目前离目标还有多远，即确定剩余的工作量。第二个问题就是项目缺少可见性的问题，对于它的讨论“人月神话”做出了“巨大贡献”。当一个项目经理或一名开发者说已经完成了80%的任务，您必须保持审慎的态度。因为剩下的20%可能还需要80%的时间，甚至永远也不能完成。您可能迟迟不能拿到可以部署的软件，对此所有人都无能为力，只能表示深深的遗憾。这确实让专业软件开发者的声誉蒙羞。但是对于大型软件开发这样的复杂工作，我们的经验确实显得有些不够。

本书向我们介绍了一种增加项目可见性、降低项目失败风险的有效实践经验。许多软件开发的资深人士认定，这种方法非常不错。我们不必把宝全部押在最后那一次“大爆炸”式的集成上，而是采用“早集成、常集成”的策略。这样做可以减少缺陷引入和缺陷发现之间的时间，提高开发效率，降低风险。您对项目报告中提到的百分比将有更大的信心，而且任何时候，您都可以得到一个可以部署的软件。虽然功能可能还没有全部实现，但它是可用的！

本书向我们揭示了这样一个道理：如果一件事很难，而您又必须做，不妨经常去做，每次做一点点。其实这也是古老的“分而治之”思想的一种应用。正所谓“滴水穿石，跬步千里”。

敏捷软件开发的许多实践都是互相关联的。持续集成在与其他实践结合时，才能将它的效用发挥到极致。本书除了介绍持续集成（Continuous Integration，CI）的基本原则和工具之外，也介绍了测试驱动、代码审查、数据库集成、信息反馈等实践和工具。人（思想）、过程和自动化工具完美结合，以形成一个和谐的开发生态环境。如果您一直在追求效率更高的软件项目管理方法，我相信本书一定能给您带来一些启发和灵感。

一本好书使您改变。它将改变您的思想，您看待问题的角度和方式，最终，它将改变您的行为。然而，所有具有重要意义的改变都不会在一夜之间发生。改变随时都在发生，但按照您的意志去领导变革却很难。如果您相信这种变革必须发生，不妨朝着这个方向去努力，经常改变，每次改变一点点。

软件业中没有银弹，不可能有某种东西在短时间内让您的开发效率提高10倍。但是我们也很容易发现不同个人和不同团队之间的开发效率相差巨大，不止10倍。那些软件高手和明星团队就像职业围棋选手，他们高得惊人的效率是多年用心改进实践的结果。

参加本书翻译工作的人员除封面署名外还有：王海燕、李国安、周建鸣、范俊、张海洲、谢伟奇、林冀、钱立强、甘莉萍。在本书的翻译过程中，我学到了很多，因此郑重地向大家推荐它。如果本书对于您改进软件开发实践有所帮助，我将十分高兴。

王海鹏

丁亥年秋于上海

Martin Fowler[⊖]序

在软件行业发展的初期，软件项目中最棘手、最紧张的时刻就是集成。能单独工作的一些模块被组装在一起，然而系统整体却常常失败，而且很难找到失败的原因。但在最近几年里，集成基本上已不再是项目中的痛苦之源，而是变成了“小事一桩”。

这种转变的关键在于更为频繁地进行集成。人们曾经认为日构建是一个较难做到的目标。但是今天我接触到的项目每天都集成许多次。很奇怪，如果您遇到很痛苦的事情，似乎一个比较好的建议就是更频繁地去做这件事。

关于持续集成，一件有趣的事情就是人们常常会对它产生的影响感到吃惊。我们经常发现人们认为它的好处不大，但它却给项目带来了完全不同的感觉。项目的可见性变得好了很多，因为问题能够更快地检测出来。因为引入缺陷和发现缺陷之间的时间变短了，缺陷的发现就更容易，您可以很容易地看看改变了什么，以便帮助您找到问题的根源。与良好的测试程序配合时，可以大大减少缺陷的数量。结果是，开发者在调试上花的时间减少了，在增加功能上花的时间更多了，他们相信自己是在一个坚实的基础上开发软件。

当然，光说应该更频繁地集成是不够的，在这个简单的词语后面有一些原则和实践，正是这些原则和实践使得持续集成变成现实。您可以找到一些建议，这些建议在一些书籍中和互联网上都会有（我很自豪，我也在这方面提供过一些内容），但是您必须亲自花力气去寻找。

所以我很高兴看到Paul把这些信息收集起来，成为一本完整的书。对于希望执行这些最佳实践的人来说，这是一本参考手册。和许多简单的实践一样，细节之中包含着许多令人烦恼的东西。在这些年来，我们已经对这些细节有了许多了解，并学会了如何进行处理。这本书汇集了这些经验，为持续集成奠定了坚实的基础，就像持续集成为软件开发奠定了坚实的基础一样。

⊖ Martin Fowler是ThoughtWorks公司的首席科学家，在面向对象分析、UML模式、软件开发方法学等方面都是世界顶级专家，著名畅销书《分析模式》、《UML精粹》和《重构》的作者。

Paul Julius序

我一直希望有人能够抽出时间来写这本书，早写比晚写好。私下里说，我总希望这个人就是我。但是我很高兴Paul、Steve和Andy最后能够一起完成这本完整的、经过深思熟虑的专著。

我一直投入在持续集成之中，做那些似乎永远也做不完的事情。2001年3月，我和朋友一起创建了开放源代码项目CruiseControl，并成为项目的管理者。在白天的工作中，我在ThoughtWorks提供咨询，利用CI的原则和工具帮助客户设计、构建和部署测试解决方案。

在CruiseControl的邮件列表中，2003年开始活动多了起来。我有机会读到几千个不同的持续集成（Continuous Integration，CI）故事。软件开发者们遇到的问题各不相同，非常复杂。开发者们完成所有这些工作的理由对我来说越来越清楚了。CI的好处，如快速反馈、快速部署和可重复的自动化测试，要远大于实现CI的麻烦。但是，在创建这类环境时却很容易忽视这一点。当我们第一次发布CruiseControl时，我从来没想到过人们会通过一些有趣的方式，利用CI来改进他们的软件开发过程。

2000年，我在一个大型的J2EE应用程序开发项目中工作，这个项目用到了J2EE规范中提供的所有功能。这个应用程序本身就已够让人吃惊，更不必说它的构建了。所谓构建，我是指编译、测试、打包并执行功能测试。Ant还处在初期，还没有成为Java应用程序构建工具的事实标准。我们使用了一套组合的shell脚本来编译所有东西并执行单元测试。我们使用另一套shell脚本将所有东西变成可部署的包。最近，我们通过一些手工的步骤来部署JAR包并执行我们的功能测试。不用说，这个过程变得费时费力，而且经常容易出错。

从那时起我就希望创建一个可重复的构建过程，只要按“一个按钮”就可以了（那是Martin Fowler的热门话题之一）。Ant解决了跨平台的构建脚本的问题。我想要的其他东西要能够处理那些烦琐的步骤：部署、功能测试以及报告结果。那时，我研究了已有的解决方案，但没有找到我想要的。在那个项目中，我从来没有找到我所希望的东西。那个应用程序成功地完成了开发并投入使用，但我知道事情还可以做得更好。

在那个项目和下一个项目之间的时间里，我找到了答案。Martin Fowler和Matt

Foemmel刚刚发布了他们关于CI的第一篇文章。很幸运，我遇到了另一些ThoughtWorks的同事，他们正致力于把Fowler/Foemmel系统变成可复用的解决方案。我很兴奋，太兴奋了！我知道这就是在上个项目中一直萦绕在我心中的问题的答案。在几周之后，我们已经准备好了所有东西，并在几个已有的项目中开始使用。我甚至拜访了一个愿意进行Beta测试的地方，在一个真正的目标企业中安装了CruiseControl的前身。对我来说，再也不会回头了。

作为ThoughtWorks的一名顾问，我遇到了一些极为复杂的企业级部署结构。我们的客户根据业界的宣传资料承诺的优势，经常希望得到快速的修复。就像所有技术一样，关于它可以怎样轻易地改变您的企业，实际上存在着一些误导。如果说我从多年的顾问工作中学到了什么，那就是没有什么事情像看起来那么简单。

我想告诉客户如何实际地应用CI的原则。我想强调从开发的“韵律”转变到真正享受到那些好处的重要性。如果开发者每个月只签入一次，不关注自动化的测试，或者并不急于修复失败的构建，那么要完全享受CI的好处就很成问题了。

这是否意味着IT经理们应该先完成向这些实践的转变，再来碰CI呢？不是的。实际上，应用CI的实践可以是促成这些改变的最快推进器。我发现，安装像CruiseControl这样的CI工具会使软件团队变得积极主动，而不是消极被动。这些改变不会在一夜中发生，您必须有正确的预期——包括那些涉及的IT经理。通过对底层原理的深入理解，即使是最复杂的环境也可以变得易于理解，易于测试，而且易于很快地投入生产使用。

本书的作者们为您整理好比赛场地。我发现这本书既全面又深入。这本书深入地讨论了CI最重要的方面，它将帮助读者做出理智的决定。书中的各种主题介绍了今天在CI领域中运用的各种方法，帮助读者衡量需要进行的折衷。最后，我很高兴看到CI社区中有这么多的工作被规范化，成为下一步创新的基础。由于这一点，我非常推荐这本书。它是一个重要的资源，利用这些CI魔法，使得企业级应用程序的复杂配置变得有意思。

前　言

在我刚刚参加工作的时候，我看到杂志上有一张整页的广告，展示了键盘上的一个键，类似Enter键，上面标着“Integrate”（集成）（参见图1）。键下面的文字是“假如一切如此容易。”我已记不清楚这个广告是谁为了什么而做的，但它打动了我的心。在软件开发方面，我曾想，这当然永远不会实现，因为在我们的项目里，我们会花几天的时间在“集成地狱”中挣扎，在接近项目里程碑的时候尝试将大量软件组件拼凑起来。但是我喜欢这个想法，所以我剪下了这张广告，把它贴在我的墙上。对我来说，它代表了我成为一名高效率的软件开发者的主要目标之一：将重复的、容易出错的过程自动化。而且，它包含我的梦想，即将软件集成变成项目中的“小事一桩”（Martin Fowler曾这样说）——只是自然发生的事情。持续集成（CI）可以帮助您将项目中的集成变成小事一桩。

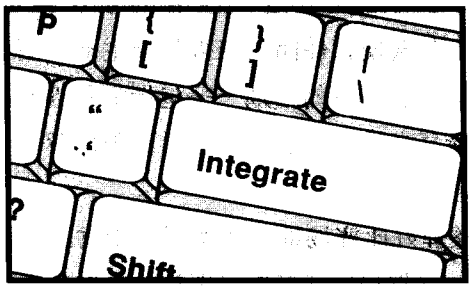


图1　集成

本书重点

请考虑软件项目中的一些典型的开发过程：编译代码、通过数据库定义数据并操作数据、进行测试、复查代码，最后部署软件。另外，团队肯定需要就软件的状态进行沟通。请想像一下如果您可以按一个键就完成这些过程。

本书向您展示了如何创建一个虚拟的集成按钮，将许多软件开发过程都自动化。而且，我们介绍了如何持续地按下这个按钮，从而减少创建可部署的应用程序时的风

险，如较晚才发现缺陷，低品质的代码等。在创建CI系统时，许多过程都被自动化，在每次修改开发的软件时，都执行这些过程。

什么是持续集成

集成软件的过程不是新问题。在一个人开发的项目中，依赖外部系统又比较少的话，软件集成不会成为太大的问题，但是随着项目复杂度的增加（即使只增加一个人），就会对集成和确保软件组件能够一起工作提出更多的要求——要早集成，常集成。等到项目快结束时才来集成会导致各种各样的软件品质问题，解决这些问题代价很大，常常会导致项目延期。CI以较小增量的方式迅速地解决这些风险。

在Martin Fowler热门的文章《Continuous Integration》（持续集成）[⊖]中，他将CI描述为：

……一种软件开发实践，即团队的成员经常集成他们的工作，通常每个成员每天至少集成一次——这导致每天发生多次集成。每次集成都通过自动化的构建（包括测试）来验证，从而尽快地检测出集成错误。许多团队发现这个过程会大大减少集成问题，让团队能够更快地开发内聚的软件。

根据我的经验，这意味着：

- 所有开发者都先要在他们自己的工作站上执行私有构建[⊖]，然后再将他们的代码提交到版本控制库中，从而确保他们的变更不会导致集成构建失败。
- 开发者每天至少向版本控制库提交一次代码。
- 集成构建每天在一台独立的计算机上进行多次。
- 每次构建都必须100%通过测试。
- 生成可以进行功能测试的产品（如WAR、配件、可执行程序等）。
- 修复失败的构建是优先级最高的事情。
- 某些开发者复查构建生成的报告，如编码标准报告和依赖分析报告，寻找可以改进的地方。

本书讨论了CI中自动化的方面，因为您会从自动化重复的、容易出错的过程中得

⊖ 参见www.martinfowler.com/articles/continuousIntegration.html。

⊖ 私有（系统）构建和集成构建模式在Stephen P. Berczuk和Brad Appleton写的《Software Configuration Management Patterns》一书中有介绍。

到许多好处。但是，正如Fowler所指出的，CI就是经常集成工作的过程，这不一定需要自动化的过程。我们相信，既然已经有许多工具让CI成为自动化的过程，那么使用CI服务器来自动化CI实践就是一种有效的方式。不管怎样，也许手工集成的方式（利用自动化的构建）很适合您的团队。

快速反馈

持续集成增加了您获得反馈信息的机会。这样，您每天都能多次了解项目的状态。CI可以用来减少引入缺陷和修复缺陷之间的时间，从而改进总体软件品质。

开发团队不应该相信因为CI系统自动化了，就可以避免集成问题。如果团队只使用自动化的工具来编译源代码，就更是如此。有些人把编译称为“构建”，实际上不是的（参见第1章）。有效的CI实践包含的东西比工具多得多。它包括本书中介绍的实践，如经常向版本控制库提交代码，立即修复失败的构建以及使用独立的集成构建计算机等。

CI的实践支持快速反馈。如果应用了有效的CI实践，您可以每天多次了解到正在开发的软件的健康状况。而且，CI与重构、测试驱动开发等实践配合得挺好，因为这些实践的中心思想都是进行小的变更。从本质上来说，CI提供了一张安全网，确保变更能够与软件的其他部分一起工作。从更高的层面上讲，CI增加了团队整体的信心，减少了项目所需的人工，因为它通常是一个无人执守的过程，在软件发生变更时执行。

关于“持续”的注释

我们在本书中使用“持续”这个术语，但是这种用法从技术上来说是不对的。“持续”意味着某事一旦启动就不会停止。这意味着集成过程一直在执行，但是即使对于最密集的CI环境来说，也不是这样的。所以，我们在这本书中讲述的更像是“经常集成”。

谁应该读这本书

在我的经验中，把软件开发当成工作的人和把它当成职业的人之间有着显著的差别。本书是为那些把软件开发当成职业，并发现自己在做一些重复的过程的人而写的（或者我们将帮助您意识到您正频繁地这样做）。我们描述CI的实践和好处，让您知道如何应用这些实践，这样您就可以将时间和专业知识用在更重要、更有挑战性的问题

上。本书介绍了与CI相关的主要话题，包括如何利用持续反馈、测试、部署、审查和数据库集成来实现CI。不论您在软件开发中承担哪种角色，您都可以将CI纳入到自己的软件开发过程之中。如果您是一位软件开发专家，希望变得更有效率（在您的时间里做更多的事情或得到更可靠的结果），您会从本书中学到很多东西。

开发者

如果您已注意到自己宁愿在为用户开发软件而不是与软件集成问题搏斗，那么本书将帮助您消除原来的大部分“痛苦”。本书不会要求您花更多的时间来集成，它是讲如何让大部分的软件集成工作变成“小事一桩”，让您能够关注最喜欢做的事情：开发软件。本书中的许多实践和例子向您展示了如何实现一个有效的CI系统。

构建/配置/发布管理

如果您的工作是让能工作的软件发布出去，您会发现这本书特别有意思，因为我们在书中展示，通过在每次对版本控制库进行变更时执行一些过程，您可以生成一致的、能工作的软件。可能许多人在管理构建的同时还承担项目中的其他角色，如开发。CI将替您完成一些“思考”，不必等到开发生命周期的末尾，它每天都能多次生成可测试的软件。

测试者

CI为软件开发提供了快速的反馈信息，消除了过去即使进行了“修复”之后，缺陷还会再次出现的痛苦。在实现了CI的项目中，测试者通常会提高满意度，并提高对他们的角色的兴趣，因为送来测试的软件更为频繁，每次要测试的范围也较小。在开发生命周期中使用CI系统之后，您的测试是一直在进行的，而不是像通常那样有时忙得要死，有时又闲着没事。在以前，要么测试者会工作到很晚，要么又没有东西可测试。

经理

对于团队一致地、重复地交付能工作软件的能力，如果您希望有更大的信心，那么本书将给您带来巨大的冲击。您可以更有效地管理时间、费用和品质，因为您的决策是基于能工作的软件以及真实的反馈信息和测量指标数据的，而不是项目进度计划

上的任务项。

本书组织

本书分为两个部分。第一部分介绍CI，从头开始讨论了CI的概念和实践。第一部分针对的是那些不熟悉CI的核心实践的读者。但是，如果没有第二部分，我们觉得这些CI的实践是不完整的。第二部分将核心概念扩展为CI系统执行的有效过程，包括测试、审查、部署和反馈等。

第一部分 CI的背景知识：原则与实践

第1章：启程，让您通过一些例子了解利用CI服务器来持续构建软件。

第2章：引入持续集成，让您熟悉常见的实践方法，知道如何开始CI。

第3章：利用CI减少风险，通过场景式的例子说明了CI可以缓解的主要风险。

第4章：针对每次变更构建软件，探讨了利用自动化的构建，在每次变更时集成软件。

第二部分 创建全功能的CI系统

第5章：持续数据库集成，进入一些更高级的概念，涉及构建数据库和应用测试数据等过程，作为每次集成构建的一部分工作。

第6章：持续测试，介绍了在每次集成构建时测试软件的概念和策略。

第7章：持续审查，介绍了利用不同的工具和技术进行自动化的、持续的审查（静态和动态分析）。

第8章：持续部署，探讨了利用CI系统部署软件的过程，以便于进行功能测试。

第9章：持续反馈，向您展示了利用持续反馈设备（如电子邮件、RSS、X10以及Ambient Orb），这样您在构建成功或失败时就能收到通知。

“尾声”探讨了CI将来的可能性。

附录

附录A：CI资源，包含了与CI有关的URL、工具和文章的列表。

附录B：评估CI工具，评估了市面上不同的CI服务器和相关的工具，讨论了它们

支持本书中描述的哪些实践，指出了每种工具的优点和不足，解释了如何利用它们的一些有趣的功能。

其他特点

本书还包括了一些其他特点，帮助您更好地理解和应用书中的内容。

- **实践**——在这本书中，我们介绍了40多个CI相关的实践。许多章的副标题就是这些实践。在多数章的开始有一张图，说明了这一章要介绍的实践，让您能够方便地寻找感兴趣的内容。例如，“使用专门的集成构建计算机”和“经常提交代码”就是本书中讨论的实践。
- **示例**——我们通过许多不同语言和平台上的例子，展示了如何应用这些实践。
- **问题**——每一章都包含了一个问题列表，帮助您评估项目中CI实践的应用情况。
- **Web站点**——本书的配套Web站点www.integratebutton.com提供了本书更新、代码示例和其他材料。

您会学到什么

通过阅读本书，您会学到一些概念和实践，它们将帮助您每天多次创建内聚的、能工作的软件。我们首先关注这些实践，然后是这些实践的应用，在可能的时候我们都提供了示例来说明。这些示例使用了不同的开发平台，如Java、.NET，甚至还有Ruby。CruiseControl（Java版和.NET版）是本书中主要使用的CI服务器，但是，我们在配套网站（www.integratebutton.com）和附录B中提供了使用其他服务器和工具的例子。

当您从头到尾阅读这本书时，您会发现以下内容：

- 实现CI如何能够做到在开发生命周期中的每一步都生成可部署的软件。
- CI怎样减少缺陷引入和缺陷被发现之间的时间，从而降低修复缺陷的成本。
- 通过经常构建软件，而不是等到开发的后期再构建软件，如何能够提高软件的品质。

本书没有介绍什么

本书没有介绍构成CI系统的全部工具——构建进度计划、编程环境、版本控制等。

它关注的是实现CI实践，从而得到一个有效的CI系统。首先讨论的是CI实践，如果书中提到的某个工具不再使用，或者不能满足您的特别需要，只要使用另一个工具来应用这些实践就行了，目的是达到同样的效果。

本书也不可能介绍CI所使用的每一种类型测试、反馈机制、自动化审查工具和部署的类型。即使这样做，用处也不大。通过关注关键实践，利用技术和工具的示例来讲解数据库集成、测试、审查和反馈。项目和团队可以根据自己的理解进行不同的应用。我们希望这样可以实现更宏大的目标。正如本书中处处提到的，本书的配套Web站点www.integratebutton.com包含使用其他工具和语言的例子，这些例子在本书中可能没有提及。

关于作者

本书有三位作者和一位撰稿者。我编写了大部分的章节。Steve Matyas参与了第4、5、7、8章和附录A的编写，并提供了本书中的一些例子。Andy Glover编写了第6、7、8章，提供了例子，并参与了本书的其他部分的编写。Eric Tavela编写了附录B。这样在读到使用第一人称的句子，您可以知道是谁在发表观点。

关于封面

当我得知我们的书将作为著名的Martin Fowler签名系列中的一本出版时，我非常兴奋。我知道这意味着我可以挑选一座桥作为这本书的封面。其他几位作者和我都是在华盛顿特区长大的。如果您不是来自这个地区，可能不知道这个地区是变化很快的。更准确地说，我们来自北弗吉尼亚，所以觉得选择弗吉尼亚的“天然桥”作为封面是很不错的。虽然我在2007年初把这座桥选为封面之后才去看过它。它有一段有趣的历史，我觉得难以置信，它竟然能每天让汽车从上面开过（当然，我也开车从上面走过几次）。我希望在阅读了本书之后，您会将CI作为下一个软件开发项目中的自然组成部分。

致谢

我说不清楚有多少次在读书时看到作者说“单靠自己是无法完成的”和其他一些

事情。我总是对自己说："他们只是在假谦虚"。可是，我确实错了。这本书是一个浩大的工程，我要感谢这里列出的人。

我要感谢我的出版商，Addison-Wesley。我特别要感谢我的责任编辑Chris Guzikowski，他和我一起度过了这个耗费心血的过程。他的经验，观点和鼓励对我帮助非常大。而且，我的项目编辑Chris Zahn，在几个版本和几轮编辑中提供了中肯的建议。我还要感谢Karen Gettman、Michelle Housley、Jessica D'Amico、Julie Nahil、Rebecca Greenberg以及我的第一位责任编辑Mary O'Brien和其他人。

Rich Mills为本书提供了CVS服务器，并在"头脑风暴"会议上提出了绝妙的想法。我也要感谢我的指导者和朋友Rob Daly，在2002年让我开始职业写作，并在我写作的过程中进行了详细的复查。John Steven对促成本书的编写也起了帮助作用。

我想感谢我的合作者、编辑和撰稿者。Steve Matyas和我度过了许多不眠之夜，创作您读到的这本书。Andy Glover是我们抓过来的作者，他提供了大量有关项目的开发者测试的经验。Lisa Porter是我们的特约编辑，她不知疲倦地检查每一个版本，进行编辑并提供建议，提高了这本书的本质。我要感谢Eric Tavela，他写了CI工具的附录，感谢Levent Gurses在附录B中提供了Maven 2的经验。

我们有一个平衡的技术评阅核心团队，他们在本书的编写过程中提供了很好的反馈意见。他们是Tom Copeland、Rob Daly、Sally Duvall、Casper Hornstrup、Joe Hunt、Erin Jackson、Joe Konior、Rich Mills、Leslie Power、David Sisk、Carl Tallis、Eric Tavela、Dan Taylor和Sajit Vasudevan。

我还要感谢Charles Murray和Cristalle Belonia的帮助以及来自Urbancode公司的Maciej Zawadzki和Eric Minick的帮助。

我要感谢几个很好的人，我在Stelligent公司工作的每一天里，他们对我提供了支持和帮助。他们是Burke Cox、Mandy Owens、David Wood和Ron Wright。还有许多人在这些年我的工作中给了我启发，他们是Rich Campbell、David Fado、Mike Fraser、Brent Gendleman、Jon Hughes、Jeff Hwang、Sherry Hwang、Sandi Kyle、Brian Lyons、Susan Mason、Brian Messer、Sandy Miller、John Newman、Marcus Owen、Chris Painter、Paulette Rogers、Mark、Simonik、Joe Stusnick和Mike Trail。

我也感谢Addison-Wesley的技术评阅团队提供的反馈信息，包括Scott Ambler、Brad Appleton、Jon Eaves、Martin Fowler、Paul Holser、Paul Julius、Kirk Knoernschild、Mike Melia、Julian Simpson、Andy Trigg、Bas Vodde、Michael Ward

和Jason Yip。

我想感谢参加了CITCON芝加哥2006年会议的人，他们和我们分享了在CI和测试方面的经验。我特别要感谢Paul Julus和Jeffrey Frederick组织了这次会议以及其他所有参加了这次会议的人。

最后，我要感谢Jenn在本书编写的日子里始终提供了坚定的支持。

Paul M. Duvall

于弗吉尼亚州费尔费克斯县

2007年3月