第一图书网, tushu007.com

<<持续集成>>

图书基本信息

书名：<<持续集成>>

13位ISBN编号：9787121148699

10位ISBN编号：7121148692

出版时间：2012-6

出版时间：电子工业出版社

作者：[美]Paul M. Duvall （保罗.M. 杜瓦尔）Steve Matyas （史蒂夫.迈耶斯） Andrew Glover(安德鲁.格洛弗) 著,王海鹏 译

页数：239

字数：435000

译者：王海鹏

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：http://www.tushu007.com

前言

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　　这本书讲什么　　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。
请想象一下，如果您可以按一个键就完成这些过程。
这本书向您展示了如何创建一个虚拟的集成按钮，将许多软件开过过程都自动化。
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的热门文章“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出一致的软件。
根据我的经验，这意味着：　　所有开发者都先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间隔，从而改进软件的总体品质。
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值守的过程，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（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。
这本书介绍了与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（在您拥有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为用户开发软件而不是与软件集成问题搏斗，那么这本书将帮助您消除原来的大部分“痛苦”。
这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。
这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，您的测试是一直进行的，而不是像通常那样有时忙得要死，有时

又闲着没事。
以前测试者要么工作到很晚，要么又没有东西可测试。
经理　　对于团队一致地、重复地交付工作软件的能力，如果您希望有更大的信心，那么这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，因为您决策的基础是能工作的软件、真实的反馈信息和测量指标数据，而不是项目进度计划上的任务项。
本书的组织结构　　本书分为两个部分。
第一部分介绍了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，这些CI的实践是不完整的。
第二部分自然地将核心概念扩展为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如何利用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，向您展示了如何利用持续反馈设备（如电子邮件、RSS、X10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。
您将学到什么　　通过阅读这本书，您将学到一些概念和实践，它们能帮助您每天多次创建一致的、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、Microsoft.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，您会有下面的发现：　　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如何能够减少缺陷引入和缺陷被发现之间的时间间隔，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系统的全部的工具--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。
正如这本书中处处提到的，这本书的配套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贡献者。
我编写了大部分章节。
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时，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。
如果你不是来自这个地区的，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。
我直到2007年初才去了那里--在我将它选为封面之后。
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）。
我希望在阅读了这本书之后，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，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、编辑和贡献者。
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

品质。
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复查者核心团队，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。
还有许多人这些年在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的技术复查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

内容概要

Jolt大奖素有“软件业之奥斯卡”的美称，本丛书精选自Jolt历届获奖图书，以植根于开发实践中的独到工程思想与杰出方法论为主要甄选方向。
本书全面深入地讨论持续集成的各个方面，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。
此外，还介绍了测试驱动、代码审查、数据库集成、信息反馈等实践和工具。
全书列举了持续集成系统的优缺点，如何去使用持续集成系统，什么时候使用等，可操作性极强。

本书荣获2008年Jolt世界图书大奖，适合软件开发人员及团队阅读，还可作为软件工程方面的教材。

作者简介

Paul M. Duvall是Stelligent公司的CTO。
Stelligent公司是一家咨询公司，他们通过优化软件开发过程，帮助开发团队可靠地、快速地开发出更好的软件。
他几乎担任过软件开发项目中的所有职务，从开发者到测试者再到架构师和项目经理。
Paul向各个行业的客户提供咨询，包括金融业、房地产业、政府、医疗卫生业，以及大型的独立软件提供商。
他是许多知名软件会议的特邀讲演者。
他为IBM developerWorks撰写了一系列的文章，名为“Automation for the People”，他是NFJS 2007 Anthology（Pragmatic Programmers，2007）的合著者，也是UML 2 Toolkit（Wiley，2003）的贡献作者。
他是临床研究数据管理系统和方法的发明者之一，这个系统和方法正在申请专利。
他经常在www.testearly.com和www.integratebutton.com上写日志。

Stephen M. Matyas III是AutomateIT的副总裁。
AutomateIT是5AM Solutions公司的一个服务机构，它帮助组织机构通过自动化来改进软件开发。
Steve在应用软件工程方面有多重背景，他的客户包括商业客户和政府客户。
Steve担任过许多种不同的角色，从业务分析师和项目经理到开发者、设计者和架构师。
他是UML 2 Toolkit（Wiley，2003）的贡献作者。
他实践了许多迭代增量式的方法，包括敏捷方法和Rational Unified Process（RUP）。
他的大部分第一手的职业经验来自于Java/J2EE定制软件开发和服务，在方法学、软件品质和过程改进方面具有特殊的要求。
他拥有弗吉尼亚理工大学的计算机科学学士学位。

Andrew Glover是Stelligent公司的总裁。
Stelligent公司是一家咨询公司，他们通过优化软件开发过程，帮助开发团队可靠地、快速地开发出更好的软件。
Andy经常在北美的各种会议上作为讲演嘉宾，他也是No Fluff Just Stuff Software Symposium小组的讲演者。
他还是Groovy in Action（Manning，2007），Java Testing Patterns（Wiley，2004）及NFJS 2006 Anthology（Pragmatic Programmers，2006）的合著者之一。
他也是一些在线文章的作者，这些文章发布在IBM的developerWorks，O’Reilly的ONJava、ONLamp和Dev2Dev门户站点上。
他经常在上写有关软件品质的日志。

贡献者简介

Lisa Porter是一个顾问团队的高级技术作者，这个团队向美国政府提供网络安全的解决方案。

Lisa在本书出版之前提供了技术编辑服务。
她早些年曾支持一个包含多个应用程序的大型软件开发项目，在需求确定和项目成熟度/能力等活动方面受到了好评。
她也进行外语翻译和架构/工程方面的技术写作。
Lisa从2002年开始就从事编辑书籍和在线出版物的工作。

Eric Tavela是5AM Solutions公司的首席架构师。
5AM
Solutions公司是一个软件开发公司，该公司致力于将软件工程的最佳实践应用于生命科学的研究工作
。
Eric的主要工作背景是实现Java/J2EE应用程序及指导面向对象软件开发和UML建模。

书籍目录

出版说明
译者序
Martin Fowler序
Paul Julius序
前言
作者简介
贡献者简介
第1部分 CI的背景知识：原则与实践
第1章 启程
1.1 针对每次变更构建软件
开发人员
版本控制库
CI服务器
构建脚本
反馈机制
集成构建计算机
1.2 CI的特征
源代码编译
数据库集成
测试
审查
部署
文档与反馈
1.3 本章小结
1.4 问题
第2章 引入持续集成
2.1 CI生活中的一天
2.2 CI的价值是什么
减少风险
减少重复过程
生成可部署的软件
增强项目的可见性
建立起更强大的产品信心
2.3 什么阻碍了团队使用CI
2.4 如何进行“持续”集成
2.5 项目应该在何时以何种方式实现CI
2.6 集成的演进
2.7 CI如何与其他开发实践配合
2.8 CI需要多少时间架设
2.9 CI与您
2.10 经常提交代码
2.11 不要提交无法构建的代码
2.12 立即修复无法集成的构建
2.13 编写自动化的开发者测试
2.14 必须通过所有测试和审查

2.15 执行私有构建
2.16 避免签出无法构建的代码
2.17 本章小结
2.18 问题
第3章 利用CI减少风险
3.1 风险：没有可部署的软件
场景：“ 在我的机器上是行的 ”
解决方案
场景：与数据库同步
解决方案
场景：点错了
解决方案
3.2 风险：很晚才发现缺陷
场景：回归测试
解决方案
场景：测试覆盖
解决方案
3.3 风险：缺少项目可见性
场景：“ 您收到了备忘录吗？ ”
解决方案
场景：不能使软件可见
解决方案
3.4 风险：低品质的软件
场景：坚持编码标准
解决方案
场景：维持架构
解决方案
场景：重复的代码
解决方案
3.5 本章小结
3.6 问题
第4章 针对每次变更构建软件
4.1 自动化构建
4.2 执行单命令构建
4.3 将构建脚本从IDE中分离
4.4 集中放置软件资产
4.5 创建一致的目录结构
4.6 让构建快速失败
4.7 针对所有环境构建
4.8 构建类型和触发机制
构建类型
私有构建
集成构建
发布构建
构建触发机制
触发构建

4.9 使用专门的集成构建计算机
4.10 使用CI服务器
4.11 执行手工集成构建
4.12 执行快速构建
收集构建测量数据
分析构建测量数据
选择并实现改进
使用专门的集成构建计算机
增强集成构建计算机的硬件能力
改进测试性能
4.13 分阶段构建
检查基础设施
优化构建过程
单独构建系统组件
改进软件审查的性能
执行分布式集成构建
重新评估
4.14 这对您如何生效
4.15 本章小结
4.16 问题
第2部分 创建全功能的CI系统
第5章 持续数据库集成
5.1 自动化数据库集成
创建数据库
操作数据库
创建一段构建数据库的结合脚本
5.2 使用本地数据库沙盒
5.3 利用版本控制库共享数据库资产
5.4 持续数据库集成
5.5 让开发者能够修改数据库
5.6 开发团队共同关注修复失败构建
5.7 让DBA成为开发团队的一员
5.8 数据库集成和集成按钮
测试
审查
部署
反馈与文档
5.9 本章小结
5.10 问题
第6章 持续测试
6.1 自动化单元测试
6.2 自动化组件测试
6.3 自动化系统测试
6.4 自动化功能测试
6.5 对开发者测试分类
6.6 先执行较快的测试
6.7 为缺陷编写测试

6.8 让组件测试可重复
6.9 将测试用例限制为一个断言
6.10 本章小结
6.11 问题
第7章 持续审查
7.1 审查与测试的区别
7.2 应该以怎样的频度执行审查
7.3 代码测量指标：历史
7.4 降低代码复杂度
7.5 持续进行设计复查
7.6 通过代码审查维持组织机构的标准
7.7 减少重复的代码
使用PMD-CPD
7.8 判断代码覆盖率
7.9 持续评估代码品质
覆盖率检查频度
覆盖率与性能
7.10 本章小结
7.11 问题
第8章 持续部署
8.1 随时随地发布可工作的软件
8.2 为库中的资产打上标签
8.3 得到干净的环境
8.4 为每一个构建版打上标签
8.5 执行所有的测试
8.6 创建构建反馈报告
8.7 回滚构建的过程能力
8.8 本章小结
8.9 问题
第9章 持续反馈
9.1 所有正确的东西
正确的信息
正确的人
正确的时间
正确的方式
9.2 使用持续反馈机制
电子邮件
SMS（文本消息）
Ambient Orb和X10设备
Windows任务条
声音
宽屏显示器
9.3 本章小结
9.4 问题
后记：CI的未来
附录A CI资源
附录B 评估CI工具

第一图书网, tushu007.com

参考文献

章节摘录

版权页： 插图： CI的实践与其他软件开发实践是互补的，如开发者测试、坚持编码标准、重构和小发行版本。
不论您是在使用RUP、XP、RUP结合XP、SCRUM、Crystal或其他任何一种方法学，都可以采用CI的实践。
下面的列表总结了CI的实践是怎样促进其他实践的。
■开发者测试：开发者编写测试时通常使用某种xUnit框架，如JUnit或NUnit。
这些测试可以通过构建脚本自动执行。
由于CI的实践鼓励每次对软件进行改动时就执行构建，而自动化测试又是构建的一部分，所以CI就使得每次对软件进行修改时，都会对全部代码执行自动化的回归测试。
■坚持编码标准：编码标准是开发者在开发项目时必须遵守的一组规范。
在许多项目中，确保遵守编码规范基本上是一个手工过程，通过代码复查来执行。
CI可以在每次发生变更时执行一段构建脚本，通过运行一些自动化的静态分析工具，根据已建立的编码标准审查源代码，报告标准的遵守情况。
■重构：根据Fowler所说的，重构是"这样一个过程，它在不改变代码的外部行为的情况下，对软件系统进行修改，改进它的内部结构。
"这使代码更容易维护，并带来其他好处。
CI可以通过在每次构建时运行审查工具，确定可能的问题，从而实现对重构的支持。
■小发行版本：这项实践让测试人员和用户能够按他们的需要的频度，得到可以工作的软件，因为软件集成每天都会进行许多次，基本上任何时候都可以得到一个发行版本。
当CI系统就位后，得到一个发行版本是不用花多少力气的。
■共同拥有代码：每个开发者都可以修改软件系统的每一个部分。
这避免了"知识孤岛"，即对于系统的特定部分，只有一个人了解相关的知识。
CI实践通过确保遵守编码标准，持续执行回归测试，对共同拥有代码提供了支持。

编辑推荐

《持续集成:软件质量改进和风险降低之道》首先从最基础的东西开始，讨论了CI的概念和实践，然后进一步讨论了CI系统执行的其他有效的过程，如数据库集成、测试、审查、部署和反馈。
通过超过40个CI相关的实践和不同语言环境下的应用示例，读者可以明白CI将导致更快速的软件开发，在开发生命周期中的每一步都能得到可部署的软件，而且减少了缺陷引入和发现之间的时间，节约了开发时间，降低了开发成本。
通过成功地实现CI，开发者可以减少风险和重复的手工操作过程，开发团队可以更好地了解项目的状态。

第一图书网, tushu007.com

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:http://www.tushu007.com