The Pragmatic Programmers

Broadview®
WWW.BROADVIEW.COM.CN

程序员修炼三部曲 第一部

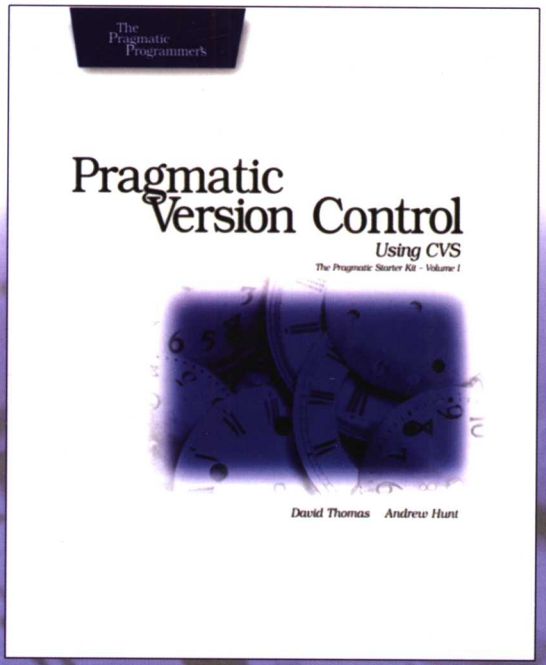
The Pragmatic Starter Kit-Volume I

版本控制之道

Pragmatic Version Control

—— 使用 CVS

Using CVS



[美] Dave Thomas Andy Hunt 著

陈伟柱 袁卫东 译

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

程序员修炼三部曲 第一部

The Pragmatic Starter Kit-Volume I

版本控制之道

—— 使用 CVS ——

Pragmatic Version Control

Using CVS

[美] Dave Thomas 著
Andy Hunt

陈伟柱 袁卫东 译

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内容简介

《**程序员修炼三部曲**》是一套由四本小册子组成的丛书，旨在帮助解决程序员在日常工作中遇到的一些具体问题的需要，内容覆盖了对于现代软件开发非常重要的基础性知识。这套丛书不仅展现了注重实效的实际技巧、工具使用，也贯穿了作者们在其名作《**程序员修炼之道：从小工到专家**》中所坚持的开发哲学。而所有这些，都是帮助开发人员和开发团队进行正常开发、不断进步，并带来高开发效率的利器。

《版本控制之道——使用 CVS》是三部曲中的第一部，它讲述如何使用版本控制给整个项目打基础，如何有效地使用版本控制系统，并从中获取最大的好处和安全性。尽管使用了版本控制可以使项目的开发工作大大提高效率，但现实中却仍有很多开发小组根本没有使用或不会正确使用版本控制。许多人抱怨版本控制过于复杂，有点望而生畏。其实他们只要掌握了如何去使用一些方便的基本用法就可以享有版本控制所带来的 90%的好处，而本书正是为了帮助读者了解这些方便的基本用法，从而比较容易地去掌握版本控制的精髓，提高开发工作的水平。

Pragmatic Version Control Using CVS by Dave Thomas, Andy Hunt, 0-9745140-0-4

版权贸易合同登记号　图字：01-2004-2484

图书在版编目（CIP）数据

版本控制之道：使用 CVS /（美）托马斯（Thomas,D.），（美）亨特（Hunt,A.）著；陈伟柱，袁卫东译. 北京：电子工业出版社，2005.4（程序员修炼三部曲；第一部）

书名原文：Pragmatic Version Control Using CVS

ISBN 7-121-01095-X

Ⅰ.版...　Ⅱ.①托...②亨...③陈...④袁...　Ⅲ.软件工具，CVS－软件开发　Ⅳ.TP311.56

中国版本图书馆 CIP 数据核字（2005）第 029738 号

责任编辑：周　筠　陈元玉
印　　刷：北京天竺颖华印刷厂
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：11.25　　字数 200 千字
印　　次：2005 年 5 月第 1 次印刷
定　　价：25.00 元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系电话：（010）68279077。质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

读者对本书的评价

这本书告诉了我许多关于如何改善 CVS 用法的建议。书中理论与实例并陈的讲解帮助我掌握了之前所不擅长的技术：如何开始使用 CVS。如果你们在十年前就编写这本书，然后让我邮购一本，那该有多好啊！

▶ **Mike Stok**：高级软件开发工程师

Exegenix Research Inc.

本书针对 CVS 新手作了一次出色的介绍。一如既往，Dave Thomas 和 Andy Hunt 是这方面最好的专家。

▶ **Andrew C. Oliver**，Apache POI 的创始人

SuperLink Software, Inc.

我已经具备了多年的 CVS 使用经验，从而愿意与你分享下面的经验之谈。CVS 不仅仅是一个非常好的工具，而且是一个对软件工业极其重要的工具。这本书把神秘的 CVS 技术带入了每个普通程序员的日常操作中。

▶ **Will Gwaltney**，软件开发测试师

SAS Institute.

这是一本优秀的书籍。对于没有接触过版本控制的任何程序员，本书为你们介绍了各种必不可少的技巧，有助于帮助你们学会如何使用版本控制，并且高效地工作。我非常喜欢本书的编写风格。它是在每个场景中讲述每一个例子与概念，而且渗入了大量的实际操作，这种写法实在是好极了。

▶ **Vinny Carpenter**，企业架构师

译序

一路走来，我们有幸翻译了 Pragmatic 系列的三本书。每本书虽然都算不上是阳春白雪，但却是下里巴人中的极品，本本都是短小精悍，内容充实，紧联实际，最能接近和帮助一线开发者。在此，感谢原作者给我们带来这一系列的精彩书籍。

贯穿本书始终，Dave Thomas 和 Andy Hunt 采用轻快易懂的写作风格，中间不乏生动诙谐的开发场景，让人读来趣味盎然。在翻译中，我们也尽最大努力保持原汁原味，使作者的睿智之言不因语言转化而略失光彩。相信您只需要花上一个周末就可以通读本书，然后发出一声感慨："wow，我终于了解 CVS 和版本控制了。"

借助这种风格，本书将带您越过 CVS 的门槛，进入它的美妙世界，踏上版本控制系统的康庄大道，帮助您提高软件开发的效率。至于书的具体内容，请读本书的第 1 章，在此不再赘述。

使用版本控制是为了给软件项目的明天买保险，而 CVS 是现今最流行的版本控制系统。本书主要面向 CVS 的初学者和对 CVS 仍然不太熟悉的日常使用者。此外，本书详细阐述了版本控制系统背后的众多运行机理，即使是使用其它版本控制系统的读者，相信阅读本书也会令他们受益良多。

本书由袁卫东和我共同翻译，卫东熟悉专业，文笔也属上乘，和他一起共同完成这本书籍的过程是一段令人愉快的经历。本书编辑陈元玉在审稿过程中提出了许多宝贵建议，同样为最后的译稿增色不少。同时希望读者在阅读过程中能够提出宝贵意见，帮助我们不断提高和改进译本。

最后，感谢周筠老师、陈英老师和本书编辑陈元玉，以及我的家人和我的爱人。在我困难的时候，是你们在我身边，在我收获的时候，我最先想到的人总是你们。

陈伟柱

2005 年 3 月 20 日 于北京

关于程序员修炼三部曲

在本系列的第一本书《The Pragmatic Programmer（程序员修炼之道：从小工到专家）》中，对现代软件开发的许多实质性话题作了概要性的介绍，并且获得了广泛的好评。自从该书 1999 年第一次印刷出版之后，就有许多读者向我们询问是否能再出一些后续书籍，或者该书的续篇。我们将会考虑此事；但首先，我们觉得有必要先提供几本基础方面的系列书籍。

在《程序员修炼之道：从小工到专家》出版之后的这些年来，我们发现：那些刚开始从事软件开发的读者，非常希望能够在软件开发的基本细节方面得到适当的指引，这样有助于他们早点养成良好的开发习惯；另一方面，那些经验丰富的读者，虽然已经能够完全理解书中的大多数内容，但是他们仍然希望在说服和指导开发小组中其它组员时，能得到一些帮助。现在，我们可以很高兴地告诉这些读者，我们已经有了几本能够给他们带来真正帮助的书籍了。

《Pragmatic Starter Kit（程序员修炼三部曲）》是一套由三本小册子组成的丛书，覆盖了对于现代软件开发非常重要的基础性知识。就内容而言，这三本小册子展现了实际操作、工具使用和开发哲学，而所有这些都是帮助开发小组正常开发、不断进步、带来高开发效率的利器。了解并掌握了这些知识之后，你和你的小组成员将能够很容易地养成好的开发习惯，并且对于你们所开发的项目而言，等于在外面织上了一层安全的网，它能让你感到安全和舒适。

第一本小册子《Pragmatic Version Control（版本控制之道）》讲述如何使用版本控制给整个项目打基础。打个比方来说，一个没有使用版本控制的项目，就像一个没有 UNDO 按钮的文字处理器：你输入的字符越多，错误所造成的伤害也就越大，因为没有撤消选择的机制。《Pragmatic Version Control（版本控制之道）》这本书将会告诉你：如何有效地使用版本控制系统，从中获取最大的好处和安全性，而不必拘泥于极端死板或者冗长可怕的过程。

第二本《Pragmatic Unit Testing（单元测试之道）》讨论如何有效地进行单元测试。单元测试是一项很重要的技术，它能够在程序员编写代码的同时，提供及时真实的反馈。遗憾的是，许多开发者对单元测试都不十分理解，没有意识到它可以令开发者的工作变得更加轻松。

第三本书《Pragmatic Automation（项目自动化之道）》[1]囊括了在代码构建、测试和发布过程的自动化方面需要的一些非常重要的实践和技术。很少有项目会因为时间过多而失败，往往都是由于时间不足所造成的；因此，《Pragmatic Automation（项目自动化之道）》主要是告诉你如何让计算机来完成更多的重复性工作，从而使程序员能更加专注于比较有趣也比较困难的工作。

就书的风格而言，这几本书和我们的第一本书一样，都是很通俗的。主要是帮助解决和满足程序员在日常工作中遇到的一些具体问题和需要。然而，这几本书并不是那种只给出一般问题、泛泛而谈的肤浅之作，它会让你充分理解这些知识，这样即使面对新出现的问题（本书可能没有明确提到的问题），你也能够根据这些知识找出自己的解决方法。

需要本书和其它书籍的更新信息，以及一些针对开发者和项目经理的相关资源，请访问我们的网站：http://www.pragmaticprogrammer.com

感谢你的阅读，并请记住要让阅读充满乐趣！

[1] 2004 年 7 月出版（编者注：中译本《项目自动化之道》预定 2005 年 5 月出版）。

前言

版本控制就像呼吸一样，当一切正常时你注意不到它起的作用，但它可以使得项目继续下去。然而，在我们去拜访遍布全世界的开发小组时，我们已经注意到一些东西：他们中的大部分没有正确地进行版本控制（很多开发小组根本没有做版本控制）。

形成这种状况有很多原因。当实行版本控制时，大部分开发小组抱怨版本控制过于复杂。他们会使用基本的版本控制命令，将文件签入到中心仓库或从中心仓库签出文件。但是当需要创建一个发布版本，或需要处理第三方代码时，他们就不会处理了。由于这种挫折，小组要么停止使用版本控制，要么在一页又一页不甚清楚的版本控制过程说明中浪费时间。

你并不需要这样做。我们在本书中说明了如何仅使用一些方便的基本用法就可以利用到版本控制所带来好处的90%。遵循这些用法，开发小组就可以立即开始享受版本控制带来的好处。

您的不断反馈对我们非常重要。如果您发现错误、遗漏或有什么建议请登录我们的网站。[2]

[2] http://www.pragmaticprogrammer.com/sk/vc/feedback.html

■ 排版标记的规范

黑体	表明这里的名词是正要被定义的名词，或者来自于其它语言的名词。
定宽字体	表明这里是方法名称，文件名称，类的名称，或者其它各种常量字符串。
	曲线箭头标记表明这些内容是比较高级的，如果你第一次没有看懂的话，可以跳过。
	“开发者 Joe”，他是我们的卡通朋友，在此他会提出一个相关的问题，你或许会发现这个问题非常有用。
-d) ⇒ Destination	命令选项助记符（这里是-d）

■ 致谢

写书的乐趣之一是去请朋友评论你的书稿。如果他们同意，那么可以视为一个惊喜。我们尤其要感谢Steve Berczuk,Vinny Carpenter, Will Gwaltney, Krista Knight, Andy Oliver, Jared Richardson和Mike Stok，他们给了我有用的评论和建议。（One of the joys of writing a book is that you get to ask friends to review the drafts. One of the surprises is that they agree to do it. We'd especially like to thank Steve Berczuk, Vinny Carpenter, Will Gwaltney, Krista Knight, Andy Oliver, Jared Richardson, and Mike Stok for all their useful comments and suggestions.）

Dave Thomas and *Andy Hunt*
September, 2003
pragprog@pragmaticprogrammer.com

目 录

关于程序员修炼三部曲 xi

前言 xiii

第 1 章 简介 1

1.1 项目中的版本控制 2

1.2 路线图 6

第 2 章 什么是版本控制 7

2.1 仓库(Repository) 7

2.2 我们应该在仓库中存放什么文件 9

2.3 工作区和操作文件 11

2.4 项目（Projects），模块（Modules）及文件（Files） 12

2.5 版本从何而来 13

2.6 标记(Tags) 15

2.7 分支（Branches） 16

2.8 合并（Merging） 18

2.9 锁选项 19

2.10 配置管理(CM) 23

第 3 章 起步 25

3.1 安装 CVS 25

3.2 创建一个仓库 30

3.3 CVS 命令 31

3.4 创建一个简单的项目 32

3.5 开始一个项目 34

3.6 进行修改 36

3.7 更新仓库 38

3.8 当发生冲突时 39

3.9 冲突解决 41

第 4 章 如何做 47

4.1 我们针对版本控制系统的基本观点 48

4.2 组织一个版本控制系统 48

第 5 章 访问仓库 51

5.1 安全性和用户账号 53

5.2 CVSROOT：目标参数串 54

5.3 设置 ssh 访问 56

5.4 使用 pserver 连接 57

第 6 章 常用的 CVS 命令 59

6.1 签出文件 59

6.2 使文件保持最新 62

6.3 添加文件和目录 65

6.4 忽略某些文件 70

6.5 重新命名文件 71

6.6 重新命名目录 73

6.7 查看修改了些什么 74

6.8 处理合并冲突 78

6.9 提交变更 82

6.10 检查变更历史 83

6.11 移除修改 86

第 7 章 使用标记和分支 89

7.1 标记、分支和做标记 90

7.2 创建一个发布分支 92

7.3 在发布分支里工作 94

7.4 生成发布版本 95

7.5 在发布分支中修复程序缺陷 97

7.6 开发人员的实验性分支 98

7.7 用实验代码工作 100

7.8 合并实验分支 100

第 8 章 创建一个项目 101
8.1 创建初始项目 102
8.2 项目的内部结构 104

第 9 章 使用模块 109
9.1 轻松划分子项目 110
9.2 CVS 模块 114
9.3 总结 120

第 10 章 第三方代码 121
10.1 带有源代码的库 124
10.2 修改第三方代码 128

附录 A CVS 用法及总结 137
A.1 CVS 命令格式 137
A.2 用法 146

附录 B 其它资源 149
B.1 在线 CVS 资源 149
B.2 其它 CVS 书籍 149
B.3 其它版本控制系统 150
B.4 参考书目 151

索引 153

第 1 章
简介
Introduction

本书将告诉你如何利用版本控制来提高软件开发过程的效率。

版本控制，有时也称为源码控制，是支撑项目开发的三大支柱之一。在我们看来，所有的项目都必须使用版本控制。

版本控制可以给开发小组和个人带来许多好处。

- 为开发小组提供了一个项目范围的**撤消按钮**：不再存在不可修改的操作，并且可以借助回滚操作来更正错误。设想你正在使用世界上最复杂的文字处理程序，它除了不具备“撤消”按钮，但具有所有你可以想到的功能。不过由于某种原因，开发人员忘记加入对**删除(DELETE)**键的支持。此时，你要多么小心翼翼地键入每一个字符，尤其当一篇大文档接近完成的时候，更须加倍地小心，因为任何一个错误都会让你不得不从头开始。在此情况下，版本控制就像是一个“撤消”按钮，它具有回到一小时、一天或一周前工作状态的能力，从而让你的开发小组敢于更快地工作，因为他们确信已经有了一个修正错误的好办法——版本控制。

- 版本控制能够帮助多个开发者以一种受控的方式对同一份代码进行操作。当开发小组中的某个成员覆盖了另一成员编辑的代码时，开发小组不会再因此而失去这部分修改信息。

- 版本控制系统记录了以前对代码所做的每次修改。如果你偶然见到某些“令人惊讶的代码”，此时如果能够借助版本控制的帮助，你就

可以轻易地找出是什么人、在什么时间以及（幸运地话）为何进行这些修改。

- 版本控制系统可以让你同时发布软件的多个版本，并且不会中断开发工作的主线。有了版本控制系统，即使是在软件发布前的**代码冻结期**内，开发小组仍然可以继续工作。

- 版本控制犹如一台项目范围的**时间机器**，让你可以设定一个日期，然后查看当时项目的确切状态。这不仅仅有助于研究的开展，而且如果客户对当前版本不满意，也可以立即回到以前的状态，重新生成以前的发布版本。

本书将从项目的角度来阐述版本控制。我们并非仅仅简单罗列版本控制系统使用的命令，而是着眼于成功项目所需的各种与版本控制相关的任务，然后说明版本控制系统如何帮助你完成这些任务。

让我们从一个小故事开始，说明版本控制实际上是如何工作的……

1.1 项目中的版本控制

Fred匆匆走进办公室，准备继续投入到新的Orinoco购书系统的工作中。（为何叫Orinoco？因为Fred的公司用河流名称来为所有的内部项目命名。）Fred冲了一杯咖啡之后，他用位于中心版本控制系统的项目最新源码更新了本地计算机上的项目源码副本。在列出已更新文件的日志中，他注意到Wilma已经修改了基础类Orders的代码。Fred开始担忧了，因为这个修改可能会影响到他的工作。但今天Wilma已经去给客户安装项目的最新版本了，在公司找不到她，也因此不能直接问她。Fred转而求助于版本控制系统，让它显示修改Orders时伴随的注释。然而遗憾的是，Wilma如下的注释根本不能给他带来任何帮助：

```
* 在Order类中增加了deliveryPreferences域（成员变量）
```

为了找出Wilma所做的修改，Fred再次求助于版本控制系统来查看

对源文件所做的修改。他注意到 Wilma 增加了一对具有初始值的实例变量，而且看起来没有任何地方能对这两个变量进行修改。这在以后很可能会成为一个问题，但并不会影响他今天的工作，所以 Fred 继续工作。

Fred 在自己的那份代码中为系统增加了一个新类和两个测试类。当创建这些文件时，他将文件的名称加入到版本控制系统中；但只有当 Fred 提交(commit)了他所做的修改时，这些文件才会真正地被加入到版本控制系统中，现在加入文件名称只是为了避免以后遗漏这些文件。

两个小时以后，Fred 完成了某个新功能的第一部分。这部分代码通过了测试，并且不会影响到系统的其他部分，所以他决定将所有这些代码签入(check in)到版本控制系统中，让开发小组的其他人也可以使用这部分功能。几年以来，Fred 发现：经常性地签入和签出(check out)代码，要比几天才进行这项工作一次更方便。因为如果文件之间发生偶然的冲突，那么针对几个文件解决冲突将会比较容易；而如果需要针对一个星期以来的修改解决这些冲突，那么将会困难很多。

■ 为何永远都不要接电话

正当 Fred 开始写下一部分代码时，电话铃响了，是 Wilma 从客户那里打过来的。听起来她正在安装的版本存在一个错误：打印的发票没有计算运费营业税。客户正在大发雷霆，并且要求马上改正这个错误。

■ 除非你使用了版本控制……

在和 Wilma 再次检查了发布版本的名称之后，Fred 从版本控制系统签出那个版本的所有文件。他将这些文件放到他的个人计算机的一个临时目录中，因为他打算用完就删除这些文件。现在他的计算机上有两份源码：主流源码和发布给客户的版本所对应的源码。因为要修改一处错误，所以他通知版本控制系统用一个标签为源代码做上一个标记。(当他修复了这处错误之后，他会加上另一个标记。这些标记的作用就像是为开发过程中重要的时间

点留下来的记号一样。通过在修改前和修改后都一致地使用命名的标记，开发小组的其他人将来就可以知道他究竟做了哪些修改。）

为了找出发生问题的代码，Fred首先写了一个测试程序。可以确定的是，当涉及运输的时候，程序根本就没有检查过营业税的计算，因为Fred的测试立即指出了这个问题。（Fred 做了一行笔记，期望能在这次的项目审核会议上提出这个问题，而且这种事情一定不能传扬出去）Fred添了一行代码，将运费加入到计税额中，然后编译、检查，而且测试通过了。为稳妥起见，他重新运行了整套测试（test suite），然后将修改后的代码签入到中心版本控制系统中。最后，他为发布分支添加了一个标记，说明已修复了错误，然后通知负责为客户提供紧急版本的QA人员。他们可以通过Fred的标记让编译系统创建一个已修正了错误的发布版本。然后Fred打电话给Wilma，告诉她修正后的版本在QA人员那里，马上就可以交给她。

做完这些之后，Fred 将发布版本的源码从自己的计算机上删除：因为修改后的代码已经签入了中心服务器，所以没有必要在本地保存这些代码。接着，他开始怀疑另一件类似的事情：在发布版本中发现的营业税错误是否同样会出现在目前的开发版本中？检查这个问题的最快方法是：将他在发布版本中写的测试代码添加到开发测试程序组中，让版本控制系统将发布版本那部分修改的测试代码合并（merge）到开发版本相应的文件中。合并过程将会对开发版本做与发布版本相同的修改。当他运行测试程序时，新的测试失败了，说明存在错误；然后他将修正的代码从发布版本移到开发版本中。（做这些工作不需要用到他的计算机中发布分支的代码，所有修改的代码都从中心版本控制系统中取得。）再一次运行了所有的测试之后，Fred 将这些修改过的代码提交给版本控制系统，这样开发小组就少了一个需要更正的错误。

危机结束了，Fred回头继续他当天的工作。在这个快乐的下午，他编写了一些测试和程序，直到下班时，才刚好编写完毕。当他工作的时候，小组中的其他成员也正在修改代码，因此他可以用版本控制系统来获取这些成果，并更新本地的源码副本。然后他运行了一次测试程序，当测试通过后，最后他签入修改的代码，准备第二天再继续工作。

■ 第二天……

不幸的是，第二天出现了意想不到的事情。昨天晚上Fred家的中央供热系统坏了。因为Fred住在明尼苏达州，并且当时是二月，所以这可不是一件小事。由于要等修理工来修理供热系统，所以Fred只能打电话向公司请一天假。

然而这并不代表他不得不停止工作。通过使用针对外网的安全连接，他可以访问办公室的网络；于是，Fred 签出最新的开发代码到他的便携式计算机上。由于在昨天回家前他已经签入了所有的代码，所以这里签出的都是最新的代码。他在家里裹着毯子坐在炉火边继续工作。在结束当天的工作前，他签入了在便携式计算机上修改的代码，这样，明天他就可以用这些代码继续工作。生活真美好（要是不用自己掏钱修理暖气的话）。

■ Story-book 项目

在Fred和Wilma的项目中，正确使用版本控制这项工作虽然并不是非常引人注意，但是却让他们可以控制源代码，并且有助于他们之间的交流，即使Wilma在几英里之外，这种交流也从未间断。而且，Fred可以查看代码的修改情况，并将修正了错误的代码应用到程序的多个版本上。另外，他们的版本控制系统支持离线工作，所以Fred的工作地点不会受到限制：当家里的暖气有故障时也可以在家工作。有了合适的版本控制系统（并且知道如何利用），对于项目中那些可能会让我们不知如何应对的紧急事件，Fred和Wilma都能够从容地处理，从来就不会手忙脚乱。