软件研发管理技术丛书

CVS

CVS 和 Nightly Build 技术

杨锦方 等 编著

清华大学出版社
http://www.tup.tsinghua.edu.cn

CVS 和 Nightly Build 技术

杨锦方　等 编著

清 华 大 学 出 版 社

(京)新登字 158 号

内 容 简 介

CVS（并行版本系统）和 Nightly Build（每晚构建）都是具有很高价值的软件研发管理技术。本书结合大量实际范例，详细介绍了 CVS 和 Nightly Build 的概念与具体实施，内容包括：CVS 概述，CVS 安装与权限配置，CVS 的工作原理，CVS 基础，在 CVS 中启动一个新项目，CVS 版本、版本标签和基线，CVS 并行开发（版本分支），CVS 协同开发，CVS 中目录文件的增删与移动，CVS 与二进制文件，CVS 安全，高级 CVS，CVS 增强工具，CVS 与 SCM，CVS 与 Nightly Build。

本书适合于软件部门经理、项目经理、设计师、工程师等从事软件研发管理的工作人员学习参考。

图书在版编目（CIP）数据

CVS 和 Nightly Build 技术/杨锦方等编著. —北京：清华大学出版社，2002.10
（软件研发管理技术丛书）
ISBN 7-302-05873-3

Ⅰ.C… Ⅱ.杨… Ⅲ.数据库系统－软件工具，CVS、Nightly Build Ⅳ.TP311.56

中国版本图书馆 CIP 数据核字（2002）第 070855 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.tsinghua.edu.cn
责任编辑：胡先福
印 刷 者：北京鑫丰华彩印有限公司
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16　**印张**：15.75　**字数**：386 千字
版　　次：2002 年 10 月第 1 版　2002 年 10 月第 1 次印刷
书　　号：ISBN 7-302-05873-3/TP・3480
印　　数：0001~4000
定　　价：25.00 元

自　序

从清华大学计算机系研究生毕业后，我去美国留学深造，之后在一家软件公司工作。在清华学习期间，美国高水平的软件工程技术引起了我的强烈兴趣。我读过不少国内出版的软件工程方面的书籍，令人遗憾的是，这些书的作者大都不是亲自参加软件开发的人士，因而作品的理论性、概念性太强，而可实践性和对实施的指导价值不足。到美国之后，我还惊讶地发现，许多已经被淘汰的软件工程技术，在国内的高等学校教科书中却不时出现。

在美国期间，我一直留意学习软件工程方面的技术和管理知识，希望通过亲身经历，学习和总结符合国内软件开发人员需要的、行之有效的软件工程技术。我参加了由公司出资的专门的软件技术培训，在实践中一直注意观察、学习和总结，并利用业余时间将一些对国内软件开发人员具有相当实用价值的软件开发技术和管理经验进行了初步整理。我希望能够把这些技术和经验带回中国，为国内的软件开发人员提供借鉴。这本《CVS 和 Nightly Build 技术》当是这些经验沉淀之中的一部分。

关于 CVS

本书的主题之一 CVS（并行版本系统）是一个有着很高价值的软件工程工具软件。权威的软件工程专家 Roger S. Pressman 在他的 *Software Engineering: a practitioner's approach* 一书中指出，软件配置管理（Software Configuration Management）工具在 CASE（计算机辅助软件工程）中处于基础地位，负责解决软件工程的基本问题——变化管理（Change Management）。他认为，软件配置管理是良好的软件项目管理的基石，是提高软件生产效率的重要手段。CVS 作为一个非常强大的版本控制系统，是实施软件配置管理的核心工具之一，能够辅助软件工程师和软件公司高效管理软件的变化——版本演进，在全球中小型软件企业中得到广泛使用，成为这些公司软件项目管理的基础平台。同时，CVS 在开放源代码世界几乎成为版本控制系统的标准，大量的开放源代码项目，如 Apache、GNU GCC、GNOME、Mozilla 等，都使用 CVS 进行管理。

CVS 能够应用于非常庞大的软件研发项目管理。我在美国供职的公司为许多大型企业提供企业级语音识别平台软件，产品覆盖了 5 个不同的主流 UNIX 平台和 Windows NT 平台，成功案例包括美国最大的航空公司美联航的自动电话定票系统、美国 AT&T 公司的移动语音业务平台、新兴券商查尔斯－施瓦布的自动语音投资系统等数百个大型应用系统。该类产品仅一个平台上的源代码就多达几十兆字节。CVS 作为其核心软件开发管理平台，从公司创业伊始，直到完成上述案例，一直在使用。这个开放源代码作品的强大威力使我极为震撼。

我也有使用非常专业的软件配置管理平台的经验，这种经验更加令我坚信 CVS 的价值。在实施完上述案例后不久，公司因为软件模块越来越多，项目越来越大，同时也由于财力逐渐雄厚，于是放弃 CVS 而转向 Rational 公司的 ClearCase 平台。我被派往 Rational

公司总部学习 ClearCase 系统的安装和使用。ClearCase 与 CVS 同样用于软件配置管理，但是功能更加强大。完成培训后，我为公司同事编写了 ClearCase 的培训教程。我对 ClearCase 的强大功能印象很深刻，但是 ClearCase 也让我认识到 CVS 是一个具有相当高水平的软件产品。CVS 与软件变动/缺陷跟踪系统集成，可以形成较为完整的软件配置管理解决方案，并且进一步提高软件研发管理的水平。

关于 Nightly Build

本书的另一个主题——Nightly Build（每晚构建，也称每日构建或日常构建）是另外一个具有极高价值的软件研发管理技术。与 CVS 不同，它不是一个软件，而是一种软件研发管理技术。全球著名的管理咨询公司麦肯锡在其 2000 年出版的 *Secrets of Software Success*（《软件业的成功奥秘》）（中文版，2001 年）中十分出色地解释了成功软件公司和失败软件公司之间的差别。在书中，他们写道“开发流程既可以毁了一家公司，也可以极大提高其生产率……尽管如此，许多软件公司却用贫乏得惊人的手段来应付这些问题……那些有着极佳机制——诸如非常清晰的团队结构、广泛的投资队伍、**日常构建**和软件再利用——的公司很大程度上减轻了程序员的挫折感。这些机制使工作更为赏心悦目。讨厌的返工和缺陷探测被削减了。同时，产品质量上升了，而且上市时间缩短了……”这段话里提到的**日常构建**十分重要，被麦肯锡定义为软件公司的成功关键因素（Key Success Factors）之一。

Nightly Build 因为其神奇威力——加快发现和改正缺陷，降低集成风险，提高终端产品质量，加强沟通与协作，大幅度缩短产品上市时间——在诞生之后迅速流行开来。微软公司 Windows NT 3.0 及其后的 Windows 系列产品没有出现像 Windows 95 那样尴尬的大幅度延期，与 Nightly Build 在微软公司的推广有很大关系。麦肯锡公司在编写 *Secrets of Software Success* 一书的时候，Nightly Build 还限于在一些成功的公司中使用，而今天，这个技术在美国已经几乎普及到一多半的软件公司和大量的开放源代码项目。如果在 www.google.com 搜索引擎上输入 nightly build 或者 daily build，能找到许多的开放源代码项目的相关链接。

关于这两项技术的实施

CVS 是相对独立的技术，我们不需要任何基础就可以应用 CVS 技术——不管原来的开发是多么混乱无序。不过，CVS 入门级的应用虽然简单，但是要在研发管理中充分发挥 CVS 的强大威力，还需要花大量的工夫去深刻理解 CVS 的方方面面。像 Nightly Build 这样的技术也可以在 CVS 的基础上实施，但需要对 CVS 有比较深刻的理解。

实施 Nightly Build 需要一定的基础：版本管理、自动构建和自动测试。对于大型软件项目，Nightly Build 有很高的难度，但是像 Windows NT 这样的产品开发也实施了 Nightly Build 技术并取得了巨大的回报，所以它仍然是可以实现的。对于中小型软件项目而言，Nightly Build 技术的实施要简单得多。根据我自己的经验，只要熟悉自动构建和自动测试技术，中小型软件的 Nightly Build 实施难度不高。

本书的写作是以指导实施为目标的，因此在讲述概念和具体命令之外，穿插了大量的实例，并且鼓励读者在例子的引导下，一步一步实践各项技术。在讲述 CVS 时，一开始先详细介绍了如何安装 CVS 和进行相应权限的配置，让所有读者在开始学习之前就能够搭建

起 CVS 操作环境，便于阅读时进行实践操作。本书同时照顾到企业实施 CVS 的需求，所举的例子也包括在协同开发环境下小组成员同时应用 CVS 的情况。对于 Nightly Build 技术也同样辅以实例进行讲解。

从作坊时代迈向工业时代

许多业内人士称中国绝大多数的软件企业都是“软件作坊”，因为管理太过无序。事实上，只要我们这些软件企业投入一定的精力去做好 CVS 和 Nightly Build 技术的实施，作坊式开发的面貌就一定会有很大改观。

但应注意的是，这两项技术不是万金油，成功实施它们只是打下了扎实的基础，并不意味着企业软件开发管理水平已经很高，因为软件开发管理的整个流程是相当复杂的，在版本管理（配置管理）之外，至少还有几个 CMM 二级的关键领域需要实施，包括项目管理和跟踪、需求管理、软件品质保证等。例如，一家公司可能在实施 CVS 和 Nightly Build 技术之后，比以前快得多也好得多地完成了一项软件开发，但是由于需求管理失败，导致软件根本无法使用。这种问题是 CVS 和 Nightly Build 技术无能为力的。要想全面提高研发管理水平，需要认真理解和执行 ISO9000 标准，或者 SEI-CMM 模型。

本书出版之前，我曾经根据手稿在北京为国内广受欢迎的自由开发组织 AKA（http://www.aka.org.cn）做有关 CVS 技术的专题讲座，参加者十分踊跃。之后，不断有公司邀请我为其员工提供 CVS 技术的培训，培训效果颇佳。这使我决定把 CVS 和一些相关技术整理成书，让更多的软件工程师有机会学习、使用 CVS 及其相关技术。

我希望本书能够成为国内对CVS和Nightly Build技术有兴趣而尚未实施这些技术的软件公司的实施指南，也希望得到软件工程师们的积极反馈，以便进一步改进，您的意见和建议请发到 jinfang@yahoo.com 信箱。谢谢！

中美两国软件开发管理的比较与启示

在美国学习和工作期间，我注意比较了中美两国软件公司在软件开发管理和技术方面的不同之处。下面以我在硅谷一家从事企业语音识别平台软件开发的公司工作的经历为例，介绍我的一些观点。

第一，软件工程师年龄和经验的反差。我工作过的是一家 NASDAQ 上市公司，公司只有 5 年时间，但是近一半的软件工程师、软件项目经理和软件设计师都不算很年轻，并有着 5 年以上的软件开发经验（不包括在校期间的经验）。其中部分软件设计师（software architect）更是年龄在 35 岁以上（有的甚至已经 45 岁），有着 15 年以上的软件开发经验。在开发过程中，这些软件设计师的经验成为公司的宝贵财富。他们在多年开发过程中积累的大量经验、教训能够让系统在设计阶段就避免许多后来让人走弯路的事情。权威的软件工程专家 Roger S. Pressman 在他的 *Software Engineering: a practitioner's approach* 一书中指出，软件质量保证体系最重要的是软件项目刚刚开始的需求和设计阶段。反观国内的软件工程师，绝大部分是刚从学校毕业不久，而且多半的软件工程师都定位在将来做管理者，这样，经验无法积累，低水平重复的现象就在所难免了。

第二，软件开发管理职位设置的差异。在国内，软件公司通常只指派一位项目经理，由他全面负责单个项目的开发和管理工作。如果项目经理不是技术高手，手下的软件工程师们就会有牢骚，如“凭什么你没我水平高，却坐着比我高的职位，拿着比我高的薪水？”这使得许多软件公司雇佣技术高手担任项目经理。他们中许多人缺乏管理方面的能力和意识，由他们领导的软件开发难以实现规范化和工程化。实际上，项目管理更多的是需要管理技能而并非开发经验，而管理水平高的人大多数并不擅长技术，更有一些管理人员原来不是受高等计算机专业教育出身，而是半路出家。在美国，我接触过的所有软件公司（或者有软件研发部门的计算机、通信技术公司）的软件项目小组都设有两个管理者的职位：软件项目经理（project manager）和软件设计师（software architect）。软件项目经理负责小组人力资源、激励、非技术方面的管理，并向上级负责。软件设计师负责项目的规划设计、技术方案选择等并为此负责。就像项目经理有一个职业发展道路，从项目经理到部门经理，再到总经理，甚至总裁一样，软件设计师也有其职业发展道路，从软件项目小组的软件设计师到软件部的软件设计师，再到公司的首席软件设计师。我想大多数人都注意到比尔·盖茨的头衔，是微软公司 chairman 和 chief software architect，也就是微软公司董事长兼首席软件设计师。可以想见，首席软件设计师的地位也是很高的。在更大规模的软件项目中，系统工程师（system engineer）和测试部门是单独设置的。

第三，开发管理应用软件水平的差异。美国绝大多数软件公司的软件开发管理应用了大量的先进软件，而国内的软件开发管理几乎是纯手工操作。本来，软件公司是为所有需要软件应用技术和解决方案的行业提供软件，其中也包括软件行业自身。具有讽刺意味的是，国内许多从事软件开发的公司为别人提供了非常有价值的软件，创造了很高的效益，却没有意识到（或者没有决心）自己也需要投资购买或者自行开发用于管理软件开发过程

的软件。许多美国公司根据自己的实际需要，开发用于辅助软件项目管理的软件，例如，著名的电信设备制造商朗讯公司的大型软件工程管理软件（Sablime）就是自己开发的；许多软件工程师自己用脚本语言写小工具，优化工作流程，提高工作效率。本书的主题 CVS 系统正是从一些软件工程师在自己工作过程中写的一些脚本程序起源的。在美国，流行的脚本语言，如 Perl、Bourn Shell/C Shell、Python、TCL/TK 等应用非常广泛；而在国内，这些语言应用十分有限。

第四，软件开发流程的差异。美国水平比较高的软件公司软件开发流程十分规范，技术文档和使用文档非常细致，量非常大。在大项目的开发过程中，各种各样的表格更是数不胜数。按照一位经理的说法，是“所有的事情都有文档记录”。不仅如此，美国更有 technical writer（技术写作师）这个职业，许多公司聘用专门的技术写作师完成部分技术文档和使用文档，尤其是给最终用户使用的使用指南一类的文档，更需要专业水平才能达到实用和易用。国内这方面的差距较大：软件工程师视文档为负担；项目经理本身是软件工程师出身，更加没有动力实施这些规范；公司老总可能又不懂软件开发管理。于是低水平现状是想躲都躲不掉了。

可喜的是，目前 CMM 认证正逐渐地在国内流行起来，有越来越多的软件公司重视流程。更重要的是，不把通过 CMM 认证作为目的，而是真正贯彻规范和经过实践证实有效的软件流程。不过，对于许多规模小、实力不强的公司而言，CMM 认证负担太重，而且不一定实用。记得有一位中型软件公司的老总对我说，“CMM 那些东西全都是文档，不管用。”这件事情可以得出三个结论：一是 CMM 这种先进管理技术真正得到认同还有一个过程，二是 CMM 的实用价值在一些管理阶段还不是太大，三是 CMM 可能的确需要进行适当的改造以适合中国的国情。

我接触过国内许多软件公司，这些公司中的绝大部分其开发管理的混乱状况让我非常吃惊。我也一直听到这样的议论，“国内绝大部分的软件公司都是软件作坊，基础不行，软件业怎么发展壮大？”我相信造成国内软件业落后的原因是多方面的，但是，我们的确需要认真思考如何补上提高软件开发管理水平这一课。

有人说，中国的软件工程技术水平低下和中国人的特点有关系。我认为这是无稽之谈，是为管理水平过于低下找借口。在国外，同样是中国人做软件工程师，同样是中国人做软件研发管理，为什么他们做得那么出色，赢得同事们的尊敬？

我接触到的中小型软件公司大部分都抱着提高开发管理水平的美好愿望，也有些公司在进行着各种努力。例如，CMM 正在一些中、大型软件企业中兴起。但是，CMM 流程模型引入的代价很高，而且在水平很低的情况下过分强调 CMM 中的模型恐怕也没有太大价值。CMM 的有效实施实际上是建立在强大的 CASE 工具基础之上的。

我在美国软件公司的工作经历告诉我，有一些非常有价值、实施起来代价相对较低的技术，能够帮助国内许许多多的软件公司走出作坊时代，进入软件开发流程化的阶段。这就是我们准备写作出版“软件研发管理技术丛书”的主要目的。真切地希望这套丛书能够为广大软件工程师和软件公司带来实实在在的变化。

杨锦方

2002 年 6 月

前　言

本书是指导软件公司或者软件开发团队实施 CVS 和 Nightly Build 技术、进行软件配置管理的操作指南，而不是讲述软件工程理论的教科书。对于软件配置管理方面的理论，读者可以参考相关的软件工程图书（如清华大学出版社出版的《软件配置管理》等）。

读者对象

本书的主要读者对象包括：

- 软件部门经理
- 软件项目经理
- 软件设计师
- 软件工程师

关于读者对象的补充说明：

软件工程师指的主要是中、高级软件工程师，初级软件工程师也可以应用 CVS 技术，大幅度提高软件开发效率。对于初级软件工程师，读懂书中 CVS 的基础部分并对 CVS 技术进行较为简单的应用并不难，但是笔者不鼓励初级软件工程师阅读软件配置管理（SCM）和 Nightly Build 的相关内容，因为这些内容的难度不适合初级软件工程师阅读。

本书导读

本书按照内容模块来组织章节，多数章节之间独立性较强，且都有导读信息引导读者。因此，对于不同的读者来说，可以有选择地进行阅读，先了解相关的基础知识和基本操作，再学习难度较大的章节。下面介绍各章节的主要内容以及针对不同读者的阅读建议。

第 1 章 CVS 概述。本章首先介绍什么是 CVS、一些实现 CVS 版本管理价值的情形和它的巨大价值所在；然后介绍 CVS 的特点、与 CVS 处于同一领域的其他产品以及选择这些工具的原则。建议读者都先阅读本章，以便对 CVS 的概况能够有所了解。

第 2 章 CVS 安装与权限配置。需要安装和配置 CVS 系统的读者应该阅读本章，而对于不承担安装与配置任务的读者，可以只读“安装 CVS 客户端”一节。本章讲述如何在 UNIX/Linux 或者 Windows NT/2000 服务器上安装 CVS 服务/客户端，并配置相应的权限。本书的所有章节都有丰富的操作示范，读者最好能够在开始的时候就配置好一个可操作的环境，以便跟着书中的例子操作学习，这样能够帮助理解新概念，提高学习效率。

第 3 章 CVS 的工作原理。本章讲述了 CVS 中十分重要的概念——客户/服务器模式、仓库和工作拷贝，并介绍了 CVS 的工作原理。如果读者没有掌握这些概念，将很难深入理解 CVS。建议所有读者都不要跳过这一章。

第 4 章 CVS 基础。本章介绍了日常使用 CVS 所涉及到的绝大部分功能。读者掌握了这一章的内容，就可以基本上没有障碍地初步应用 CVS 系统。要深入理解并掌握 CVS 和它的多种开发模式，则需要在熟悉本章以后阅读随后的章节。这一章让喜欢通过具体操作

来学习的读者能够迅速开始一些基本的操作。

第 5 章在 CVS 中启动一个新项目。本章介绍了如何在 CVS 中启动一个新项目，讲述了 CVS 的重要概念——模块，以及如何建立项目与 CVS 模块的联系，并详述了如何设置项目的权限以防止混乱。

第 6 章 CVS 版本、版本标签和基线。本章讲述了 CVS 的重要概念——版本和版本标签，同时还介绍了基线（baseline）版本这一重要的软件开发管理概念以及如何用 CVS 版本标签设定基线版本。

第 7 章 CVS 并行开发（版本分支）。本章讲述了 CVS 并行开发的利器——版本分支。版本分支是一项高级功能，对于初学者来说并不重要，所以这一章可以留待读者熟悉 CVS 之后再学习。本章介绍了何时需要版本分支，如何创建和使用版本分支，应该如何小心应用版本分支，以及如何将多个分支合并起来。

第 8 章 CVS 协同开发。本章介绍了第 4 章 CVS 基础中讲述的协同开发问题之外的一些协同开发的基础知识——协同开发中的状态和同步。

第 9 章 CVS 中目录文件的增删与移动。本章讲述如何在 CVS 管理的项目中添加、删除目录和文件，以及如何移动（或重命名）目录和文件。这一章的内容简单易懂，新概念很少，可以很快读完，或者在初学时跳过，在需要的时候再来参考。

第 10 章 CVS 与二进制文件。本章介绍如何在 CVS 中对二进制文件进行版本管理。这一章的内容简单，而且相对独立，读者可以在需要的时候用作参考。

第 11 章 CVS 安全。本章介绍了 CVS 安全的重要性，以及如何加强 CVS 安全性。

第 12 章高级 CVS。本章介绍了如何使用各种 CVS 配置文件，如何配置 CVS 的触发器程序辅助协同开发中的通信，使得 CVS 在一个开发者进行提交、贴标签操作时向相关人员发出电子邮件，或者向新闻组发布消息，或者向变动/缺陷跟踪系统发布变动/缺陷跟踪信息，进行完整的软件配置管理。此外，本章还介绍了 CVS 仓库管理、CVS 仓库历史信息等内容。

第 13 章 CVS 增强工具。本章介绍了几种常用的 CVS 辅助工具。

第 14 章 CVS 与 SCM。本章讲述了软件配置管理的概念、流程以及如何在 CVS 的基础上实施软件配置管理。

第 15 章 CVS 与 Nightly Build。本章讲述了提高软件开发效率最有效的利器之一——Nightly Build（每晚构建），详述了它的流程和益处，以及如何在 CVS 的基础上实施这一技术。本章还介绍了一个实际项目中的 Nightly Build 脚本程序例子。

其他说明

本书的操作例子主要以 UNIX/Linux 平台的操作环境为基础。因为 CVS 在 UNIX/Linux 和 Windows 平台上的操作原理和操作命令完全相同，所以这些例子在 Windows 平台上同样有效，只不过换为图形用户界面罢了。如果你的公司主要在 Windows 平台上进行开发，这些例子同样能够引导你和你的同事们使用 WinCVS 1.2 或者 CVS for Win32X86 这些基于 Windows 的客户端软件。

本书所举的例子全部是针对开发人员用的源代码文件。实际上，CVS 能够对任何文件进行版本管理，CVS 的全部功能对所有类型的文本文件都有效。所以，不仅可以用 CVS

来管理软件开发，还可以用来管理文本类型的文档，例如 HTML、XML 文件等。

书中绝大部分例子都是在一个 Linux 客户计算机上进行的操作。这个 Linux 客户机的主机名是 hpe800，试验用的 UNIX 账号名字是 jyang，用户 jyang 的主目录（HOME Directory）是/home/jyang。

书中对各类英文的字体做了如下规定：

- 计算机回显内容使用常规 Courier New 字体，如：`computer display`
- 用户输入内容用 Courier New 字体加黑表示，如：`[jyang@hpe800 jyang]$ user input`
- 变量用 Courier New 字体的斜体表示，如：*variable*
- 其他英文内容用 Times New Roman 字体表示，如：Times New Roman

虽然学会使用 CVS 并不是一件太难的事情，但是，要深入理解 CVS、利用 CVS 进行软件研发流程的高效管理、应用 Nightly Build 技术大幅度提高软件开发的效率并非易事。为了方便读者获取 CVS 的有关资料，我们在 http://www.unitedinfo.com.cn/CVS 网站上开设了 CVS 专栏，其中有 CVS 问题解答的电子公告牌，读者在阅读本书的过程中如有任何问题，可以到 CVS 专栏电子公告牌上提问，或发邮件至 jinfang@yahoo.com，我们将尽可能地为您解答各种与 CVS 和 Nightly Build 技术相关的问题。

本书写作过程中得到了很多朋友的支持帮助，在此对他们表示诚挚的感谢。限于经验和水平，书中不足和纰漏之处在所难免，恳请广大读者批评指正。

作 者

2002 年 6 月

目　录

第 3 篇 CVS 高级篇

第 4 篇 Nightly Build 技术

第1篇 CVS入门篇

第1章 CVS概述

1.1 CVS是什么

CVS是Concurrent Versions System的缩写，翻译成中文，也就是并行版本系统。CVS是开放源代码软件世界的杰作，是一个强大而复杂的现代版本控制系统，而版本控制系统是计算机辅助软件工程的基础平台。

对于软件开发者而言，源代码是他最重要的智力成果；对于软件公司而言，源代码是最重要的资源和资产之一。CVS是保护这些资源，并对其进行有序管理的强大工具。没有版本控制，软件研发的管理将处于一种作坊式的无序状态。

虽然在开放源代码世界中，CVS曾经有着众多的竞争者，例如SCCS（Source Code Control System）、RCS（Revision Control System）等，但是今天CVS已经成为开放源代码组织使用的标准版本控制系统，它在几乎所有的开放源代码项目中得到应用。著名的例子包括全球最流行的Web服务器Apache、Linux平台上流行的桌面系统GNOME、世界上最流行的UNIX平台C/C++编译器GCC和功能强大、应用广泛的FreeBSD、OpenBSD操作系统等。以FreeBSD为例，其CVS管理的代码超过400兆字节，文件数超过2万个，有超过50个的开发人员在CVS的基础上工作（注：数字由FreeBSD提供，来自www.cvshome.org网站）。

在著名的开放源代码项目开发平台网站http://www.sourceforge.net上，有着数以万计的软件项目在进行开发，许多原来分散的开放源代码项目都被移植到这个平台上。登录这个网站，你会发现每个项目中都有一个“CVS”链接，因为CVS是这个巨大的开发平台的标准版本控制系统。

此外，CVS因为其简单易用，功能强大，在全球中小型软件企业中得到广泛使用，而且正在继续扩张它的领地。笔者工作过的一家美国软件公司在规模达到400人的时候，仍然在使用CVS作为他们的庞大软件系统的版本管理系统。

国内现在也有部分公司将CVS作为自己的版本管理平台，如著名的互联网络和电信软件供应商亚信公司（AsiaInfo）、神州数码金融软件公司等。相信大部分国内软件项目规模都不如或者不比上面这些例子中的项目大太多，所以可以放心采用CVS。