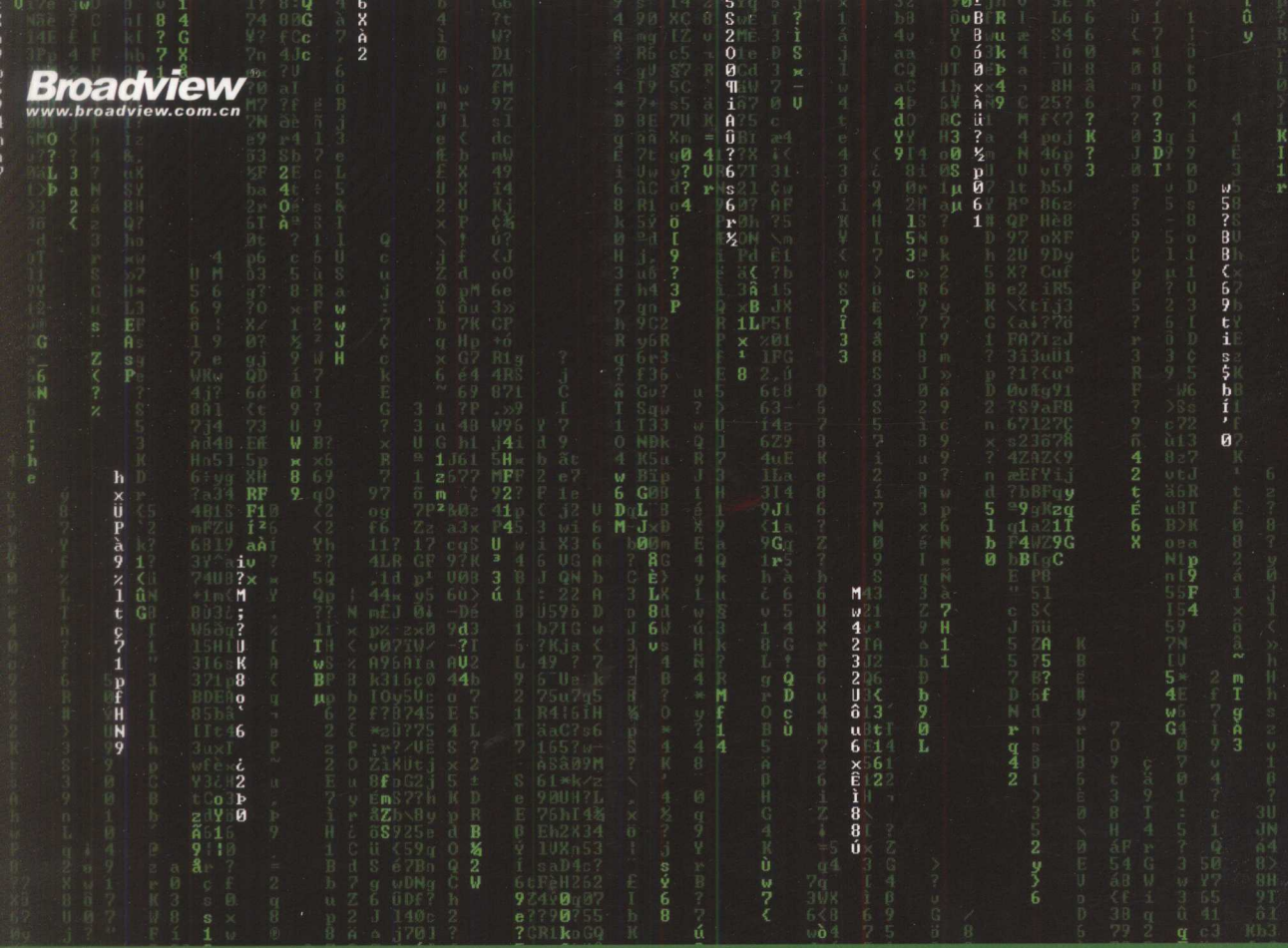

代码管理
核心技术及实践

刘冉 肖然 覃宇 ◎著

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

作者简介

刘冉，资深软件质量咨询师，拥有超过13年的软件开发和测试工作经验，熟悉自动化测试系统开发及敏捷中的QA，深入理解软件测试及SCM、CI。现在关注软件测试全自动化和敏捷中的QA，以及如何帮助大型团队有效地管理代码和CI，其中包括如何通过有效的代码分支管理、代码提交及CD保证和改进软件的质量。

肖然，精益敏捷专家，在过去15年的从业经历中，先后从事了算法复杂度研究、工业软件开发、全球项目管理，以及大型企业转型等工作。始终把软件开发作为自己的爱好，在各大企业和社区宣扬精益和敏捷的开发思想，践行有高响应力的开发理念。

覃宇，高级软件咨询师，拥有超过10年的移动应用开发经验，为Android技术专家、Git资深用户和狂热爱好者、“主干开发”的坚定拥护者和实践者，曾帮助多个客户团队改进代码管理、依赖管理、分支策略、持续集成等技术实践。

代码管理
核心技术及实践

刘冉 肖然 覃宇 ◎著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书首先通过系统化的介绍和比较，从整体上讲解了代码管理工具和系统的历史和发展。其次分别从小型团队、中大型团队、分布式大团队、基于微服务的团队及开源团队的角度总结了代码管理的核心技术及实践经验，其中包括不同类型的团队对代码管理工具和系统的选择，以及代码管理的流程、策略和技巧，还有一些代码管理工具和系统的难点、痛点等，包括如何选择分支策略、如何管理多产品线的代码、代码备份策略，以及如何在大型团队中将代码从 Subversion 迁移到 Git 等。本书可帮助读者在现实中从团队的大小及代码管理模式是集中式还是分布式、开源还是闭源等各个角度去了解和思考代码管理的核心技术和实践经验，从而帮助团队建立起一套高效的代码管理系统、策略和流程。

本书的读者对象主要是每天都需要使用代码管理工具的程序员、代码管理工具和系统的管理人员，以及团队的技术领导人员。

图书在版编目（CIP）数据

代码管理核心技术及实践 / 刘冉，肖然，覃宇著. —北京：电子工业出版社，2018.1
ISBN 978-7-121-32849-7

Ⅰ. ①代… Ⅱ. ①刘… ②肖… ③覃… Ⅲ. ①软件开发 Ⅳ. ①TP311.52

中国版本图书馆 CIP 数据核字（2017）第 247838 号

策划编辑：董　英
责任编辑：徐津平
印　　刷：三河市双峰印刷装订有限公司
装　　订：三河市双峰印刷装订有限公司
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
开　　本：787×980　1/16　　印张：13.75　　字数：251 千字
版　　次：2018 年 1 月第 1 版
印　　次：2018 年 1 月第 1 次印刷
印　　数：2500 册　　定价：59.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

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。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

前　言

我从 2004 年开始直到现在都在从事软件开发工作，经历了没有代码版本管理、代码集中式管理，以及现在的分布式代码管理，在这一过程中，我深刻体会到代码管理在软件开发中的重要性。近几年，随着软件开发规模越来越大，开发团队的规模也随之扩大，出现了越来越多的分布式团队，工程效率问题也越来越突出，比如 QCon 在 2016 年首次举办了“工程效率提升”专题。由此可见工程效率已经成为现代软件业中一个无法让人忽视的问题。

在工程效率这个范畴里，代码管理占据了举足轻重的地位，因为代码是开发人员每天工作的主要对象和内容，如果不能有效地管理，必然会影响开发人员的工作效率。随着团队规模的扩大，代码管理对团队工程效率的影响也越来越大。而高效的代码管理就像一根纽带，把所有程序员有效地串联起来，让程序员可以更高效地协同开发、编写代码，完成软件的开发工作。我们在咨询工作中遇到的很多客户都对使用代码版本管理有各种问题和困惑。出于以上原因，我们觉得有必要基于经验写一本代码管理实践相关的图书。鉴于时间有限，在本书中我们只选择了自己认为核心的技术及实践。

本书首先通过系统化的介绍和比较，让读者从整体上系统地了解代码管理工具的历史和发展。然后分别从小型团队、中大型团队、分布式大团队、基于微服务的团队及开源团队的角度，总结了代码管理的核心技术及实践，其中包括不同类型的团队对代码管理工具的选择、代码管理的流程、策略和技巧，以及一些代码管理工具和系统的难点和痛点等，可帮助读者在现实中从团队规模的大小、集中式还是分布式、开源还是闭源等角度去了解和思考代码管理的实践经验。

全书共分 3 部分，其中第 1 部分主要系统化地介绍了代码管理的历史和分类，列举并简单比较了业界常用的各种代码管理工具和系统，以及迁移工具等基础知识，以帮助读者更好地选择代码管理工具。主要以集中式代码管理工具 Subversion 为主，并以一个虚拟小团队的工作流程介绍小团队的代码管理实践，最后总结了我们经历过的传统中大型团队的代码管理的核心技术及实践。第 2 部分以介绍当前流行的分布式代码工具 Git 为主，结合大型软件项目和分布式开发团队介绍了当前流行的分布式软件开发中代码管理的核心技术及实践。第 3 部分主要介绍了正在兴起的微服架构下的代码管理实践，以及一种越来越重要的软件开发模式：开源模式下的各种代码管理核心技术及实践。

阅读提示：本书不是介绍代码管理工具的专业书籍，所以不会对书中提到的代码管理工具或系统进行全面性和系统性的介绍，所以读者需要对书中提到的代码管理工具或系统全面和深入地进行学习，并阅读与其对应的专业书籍，比如 Subversion 的 *Version Control with Subversion*、Git 的 *Pro Git* 等。如果读者来自一个大型团队，则可以略过第 2 章的独立小团队的内容，在剩下的章节中找到有用的知识点。如果读者来自一个小型团队，那么可以将第 3、4、5 章作为兴趣阅读，但是在尝试里面的一些核心技术和实践之前一定要认真思考，因为它们很可能并不适应读者现在的团队环境和规模。它们更像是一把双刃剑，所以不妨将这些内容作为未来团队扩张之前的知识储备。

书中难免存在一些错误和不妥之处，敬请谅解并欢迎指出，我们将及时修改并发表在勘误中，谢谢。

刘冉

2017 年 10 月 12 号写于成都

读者服务

轻松注册成为博文视点社区用户（www.broadview.com.cn），扫码直达本书页面。

- **下载资源**：本书如提供示例代码及资源文件，均可在 下载资源 处下载。
- **提交勘误**：您对书中内容的修改意见可在 提交勘误 处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **交流互动**：在页面下方 读者评论 处留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：http://www.broadview.com.cn/32849

目　录

第 1 部分　基础与传统

第 2 部分　当前与流行

第 3 部分　发展与未来

第 1 部分

基础与传统

第 1 章

代码版本管理工具与系统

1.1 引言

自从 20 世纪中叶有了现代软件编程语言以来，对软件代码的管理一直是软件开发中的一个不大不小的问题。随着软件规模的不断扩大，开发人员也不断增加，特别是 20 世纪末互联网的爆发及 21 世纪初移动互联网的崛起，使得软件系统开发已经从相对独立的开发模型变为一个复杂的开发模型，比如需要多种语言、使用大量的第三方代码库、出现大规模的分布式团队等，导致代码管理出现了类似于 20 世纪软件危机的代码管理危机。为了解决这种代码管理危机，近几十年以来，不同的软件公司或者个人做了各种努力，开发了各种代码管理工具和系统，用于解决在不同的时代和情况下遇到的各种代码管理问题，比如著名的 Subversion 和 Git 就是为了解决不同的问题而产生的。

1.2　代码版本管理工具的历史

1.2.1　第 1 代：本地代码管理

在 20 世纪中叶，由于软件开发相对封闭和高端，只有少量大型企业、政府和学校拥有昂贵的计算机，能使用计算机进行编码的人相对较少，软件规模也不大（相对于现在而言），代码数量也不多，所以软件代码一般以一台计算机为单位，并且直接存储在计算机系统本地。由于一个软件系统的开发周期比较长，并且有可能由多人分别开发，对于代码更改的日志跟踪与回归、代码的锁定和合并都有需求，所以在那个时代就出现了 SCCS（1972 二年）、RCS（1982 二年）等基于计算机系统本地的代码管理工具。但是随着软件规模的扩大，软件开发人数逐渐增多，本地代码管理工具已经无法适应大规模的软件开发，所以出现了第 2 代基于服务器-客户端模型的代码管理工具。

1.2.2　第 2 代：中心服务器代码管理

20 世纪七八十年代，软件开发飞速发展，特别是 PC 的出现和普及，现代语言如 C（1972 二年）、SQL（1978 二年）、C++（1983 二年）、Lisp（1984 二年）、Erlang（1986 二年）等的涌现，以及信息化革命中对大量信息系统软件的需求，推动了大量的商业软件公司和开源软件组织的出现，使得软件开发成为一个规模化的社会活动。而规模化给软件开发带来了诸多问题，其中一个就是如何在规模化的情况下管理软件代码。为了解决这个问题，一些开源组织和商业软件公司开发了第 2 代代码管理工具和系统，比如 CVS（1986 二年）、ClearCase（1992 二年）、Visual SourceSafe（1994 二年）、Perforce（1995 二年）、Subversion（2000 二年）等。

这一代代码管理工具和系统的主要特点是基于中心服务器端和客户端，软件开发者通过客户端从服务器端获取代码，并将自己的代码通过客户端提交到服务器端进行存储，代码服务器端会记录所有的代码提交日志并供开发者获取和查看。多个开发者可以同时获取同一个服务器上的代码，并向这个服务器提交代码，如果遇到冲突，就需要自己修复冲突

后再进行提交。

虽然这一代中心服务器端代码管理工具有以上优点，并且解决了规模化软件开发带来的一些问题，但是仍然存在不少限制，如下所述。

◎ 在无法连接服务器的情况下，无法查看以前提交的日志和比较代码版本（在网络十分缓慢的情况下工作也十分困难），很多开发者无法完成正常的代码开发工作。

◎ 不支持本地分支策略，导致开发分支创建缓慢，分支管理困难，并且一旦创建就很难修改，不适用于快速迭代开发流程。

◎ 由于一般只有一个中心服务器端，一旦发生灾难性问题，所有日志都会丢失，所以需要经常备份（有些大型公司在有条件的情况下自己实现了高可用的代码服务器来做灾备）。

◎ 如果软件代码量过于庞大，则可能出现速度缓慢的情况，因为每次查询、比较和提交等操作都需要和服务器通信，会对代码服务器造成很大的负载。

1.2.3 第 3 代：分布式代码管理

20 世纪 90 年代和 21 世纪初，随着大规模开源软件系统、互联网及智能移动开发的出现，软件开发进入了一个新纪元，其特点如下。

◎ 规模越来越大，代码量与代码提交量巨大。

◎ 开发人员越来越多，可能分布在不同的城市，甚至不同的国家。

◎ 组成部分越来越复杂，软件系统开发会使用多种不同的语言、技术及开发模型。

◎ 随着软件开发方法的普及（比如 Scrum、XP、Lean 等），人们需要对软件代码版本和分支更加灵活与方便地进行管理。

为了解决这些问题，出现了第 3 代分布式代码管理工具，比如 Git（2005 年）和 Mercurial（2005 年）。第 3 代代码管理工具结合了第 1 代和第 2 代的优点并实现了分布式的代码版本

管理，其特点如下。

◎　在没有和服务器连接的情况下仍然可以查看日志、提交代码和创建分支。

◎　支持本地代码分支，可以快速方便地实现各种分支管理。

◎　对代码可以进行分布式管理，从而可以实现分代码库管理的负载分流。

当然，当前这些工具还存在一些局限，相对于之前的代码管理工具学习曲线较高，管理理念与前两代区别较大，其中包括如下内容。

◎　代码基于分布式的方式进行管理。

◎　灵活的分支策略，包括轻量的本地分支与远端分支。

◎　灵活的代码版本管理。

◎　较复杂的代码同步策略。

1.3　常用的代码管理工具

代码管理工具是代码管理的基础，不同的管理工具有不同的特点、局限、成本及理念，而且对于一个团队或者公司来讲，一般选择了一种代码管理工具就意味着已经选择了这种代码管理的理念和体系，并且使用的时间越久就越难以更换，所以需要充分了解不同的工具，从而能够选择适合自己团队或者公司的代码管理工具。

1.3.1　Perforce

Perforce 是 20 世纪相关行业中认可度最高的第 2 代商用代码管理工具，其主要客户包括 Google、微软等大型软件公司（虽然 Google 和微软在代码量超过 Perforce 的管理极限时自己重新开发了专属的代码管理系统，但是也深受 Perforce 的影响，直到现在还有 Google 员工宣称 Google 使用和 Perforce 一样的单一中心代码库）。但是 Perforce 价格昂贵，中小