版本控制工具在软件开发项目管理中的应用——以 GIT 为例

王真
（北京邮电大学经济管理学院，北京 100089）

摘要： 版本控制工具的应用规范了软件开发流程，为开发团队提供了便捷的沟通方式，对提高软件开发项目质量起到了重要作用。基于对版本控制工具 GIT 的剖析与解读，阐述版本控制对软件开发项目管理的关键作用，着重分析 GIT 的工作原理及其实际应用，并通过分析 GIT 的技术优势及 GitHub 协同开发阐述版本控制工具在软件项目管理中的意义与价值。

关键词： 版本控制；项目管理；GIT；GitHub

0 引言

项目管理是运用相关管理知识、技能与工具，实施项目活动中的计划、组织、领导、控制等工作，保障整个项目在有限的资源及时间约束下实现既定目标。软件项目管理则是以软件为产品的项目管理，但软件商品特定的性质（无形性、无损耗性和易复制性）导致软件项目与一般的工程项目相比具有特殊性，而这种特殊性增加了管理的难度。

软件开发项目管理需要对软件需求、项目成本、项目进度、项目风险、配置和资源和软件开发质量进行全面管理。在软件开发过程中，除了产出的代码文件、源程序外，还将生成需求计划表、产品文档、用户手册、API 文档、各种支持库等工件，而且这些工件会随着项目的推进不断变更，形成一个数量较大、种类繁多且更改频繁的工件库。面对这样一个复杂的工件库，如何完成对工件的有序管理、快捷查找、高效利用，成为制约软件项目开发质量的一个关键问题[1]。

1 版本控制概述

软件开发的质量决定了软件项目验收的质量，而版本控制工具对有效提高软件开发质量具有不可替代的作用。在软件开发过程中忽视版本控制，将引发整合、上线、测试等方面的一系列问题，如代码不一致、文档内容冗余、过程并发处理不及时、信息安全无法保障等。

版本控制的任务是对特定目标在不同时期表现出的状态进行记录和维护，根据实际应用背景选择合适的版本间的拓扑结构，至少应实现以下功能：生成新版本、保证个版本间的协调独立、有效记录不同版本的状态、实现版本切换后对象的映象和指定的版本保持一致。版本控制的作用主要表现在两方面：一是有助于实现并行开发，提高效率；二是能够有效记录在不同时期软件开发过程中产生的各类工件，保证在需要时能够回溯至上一版本，减少由于工件丢失、修改记录丢失以及相互覆盖的损失，达到有效保护项目数字资产的目的。

常用的版本控制系统有很多，不同版本控制系统适用不同的工具。本文将版本控制系统分为三类，具体见表 1。

表 1 版本控制系统分类

名称	原理	代表工具
本地版本控制系统	采用一种简单的数据库记录文件在不同时期更新的差异	RCS
集中化的版本控制系统	项目中的开发者通过特定的客户端连接到中心服务器，进行代码拉取或提交更新操作	CVS，Subversion，Perforce
分布式版本控制系统	无中心服务器，项目中每名开发者均拥有版本库	GIT，Mercurial，Bazaar，Darcs

本地版本控制系统功能单一，很难实现多人协同开发，在如今的项目中很少使用。在实际的开发环境中，软件开发项目常以多人协作的方式开发，集中式版本控制系统解决了协同工作的难题，但是所有版本均保存在中心服务器，一旦中心服务器发生文件丢失将无法修复，面临巨大的开发风险。在这种背景下，无中心服务器的分布式版本控制系统应运而生。使用分布式版本控制系统的每个协同开发者都拥有一个完整的版本库，无论开发者中的哪一台服务器发生故障，事后都可以用其他系统中的开发者的本地仓库恢复。显然，分布式版本控制系统在安全方面优于集中式版本控制系统，已成为项目管理中版本控制的优选方案。其中，由 Linux 团队提供开发的开源分布式版本控制系统工具 GIT，目前受到众多项目团队的青睐。

2 GIT 的工作原理和功能特点

作为当前较受个人以及团队项目青睐的版本控制工具，GIT 带有的开源、分布式特性使其可以高效、敏捷地完成对各种项目的版本控制[2]。

2.1 工作原理

GIT 的工作原理见表 2。

表 2 GIT 工作原理图例

图例	解释
Workspace	工作区间，即创建的工程文件
Index	又称暂存区，提交代码、解决冲突的中转站
Localrepository	本地仓库，连接本地代码跟远程代码
Remoterepository	远程仓库，即保存代码的服务器，如 GitHub 网站

如图 1 所示，上方标示的增加（add）、提交（commit）、推送（push）等操作展示了 GIT 仓库的建立过程，实现了代码整合。同时，GIT 支持用户从远程历史仓库中拉取本地仓库的最新版本，克隆（clone）到本地，实现任何代码目标或代码仓库的任意阶段状态的“回滚”。

从底层原理来看，GIT 实际上是一个键值对（key-value）文件数据库，存储文件时将返回一个哈希值（文件 + 头部信息进行 sha-1 校验得到的校验和，取前 40 位）作为键值（key），通过该键值可检索到相应的文件内容，也就是值（value）。

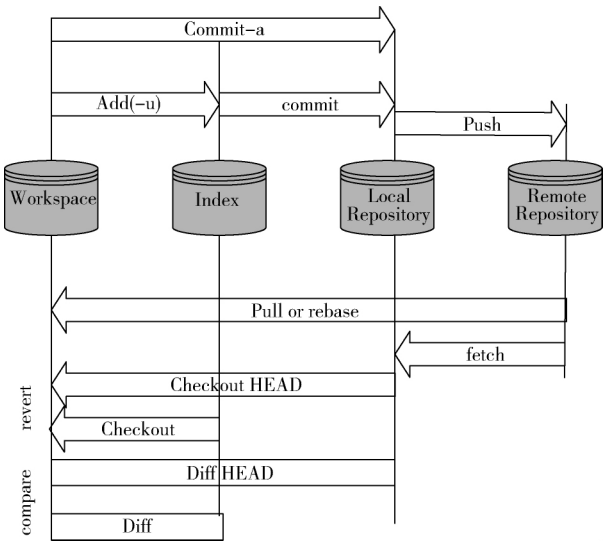

图 1 GIT 工作原理图解

GIT 对象的存储方式如以下公式所示

$$Key = sha1(file\_header + file\_content)$$ [3]

$$Value = zlib(file\_content)$$ [3]

GIT 将文件头部信息与原始数据连接起来构成一个新内容，将新内容的 40 位校验和的头部两位命名为 object 目录中的子目录，将后 38 位作为子目录中的子文件名；GIT 采用 zlib 的方式压缩数据内容，最后将已压缩的内容写入磁盘，完成对象的存储。

2.2 功能特点

2.2.1 版本库的本地化

实际上，使用 GIT 的项目，其绝大多数资源并不是存储在中心服务器，而是存储在开发者的本地仓库中。团队中的协同开发者在对执行代码的查询、更新、删除等进行操作时只需要操作本地仓库，不影响远程服务器端。协同开发者的更新可以选择在本地测试通过且当前网络环境通畅、工作无干扰的情况下提交至线上，不再受网络限制。

2.2.2 分支处理

使用 GIT 分支可以将开发任务从开发主线上分离开来，在不影响主线的同时完成开发任务[4]。在 GIT 中，常用的分支如下:

(1) Master：主分支。反映源代码整合完毕的状态，用来发布主要版本。

(2) Develop：日常开发分支。保存各协同开发者的最新代码，最简单的协作开发模式便是使用develop分支用来开发功能，开发完成并且测试通过后，将develop分支的代码合并到master分支并进行发布，实现版本上线。

(3) Feature：开发具体的功能。一般存在于协同开发者本地，项目期间使用GIT rebase保证与develop分支的代码一致并解决代码矛盾。

(4) Release：预发布版本分支。

(5) Hotfix：修复线上bug。当线上代码出现bug时，则需要开一个hotfix分支，在本地对代码进行修复后再将hotfix分支合并到develop分支与master分支并进行发布。

2.2.3　技术优势

除上述两个功能特性以外，GIT与集中化的版本控制系统的代表工具SVN相比仍具有很多优势[5]，部分比较项见表3。

表3　GIT与SVN对比

比较项	GIT	SVN
模式	分布式	集中式
存储	按元数据存储方式	按文件
全局版本号	无	有
内容完整性	Sha-1算法保证内容完整性	无
版本库	无限个	一个中央版本库
网络依赖	弱	强
克隆速度	快	慢
中央服务器	无，开发人员使用Local Repository	有，脱离中央服务器无法工作
提交	提交是本地操作	提交会被直接记录到版本库
分支操作	分支操作不会影响其他开发人员	创建新的分支影响其他开发人员

3　GIT应用-GitHub协同开发

GitHub是一个软件开发项目的代码托管平台，同时面向开源软件项目及私有软件项目，只支持GIT的版本库形式，目前支持GIT代码仓库托管、Web管理界面、订阅、讨论组、文本渲染、在线文件编辑器、协作图谱（报表）、代码片段分享（Gist）等功能。

使用GitHub可以远程管理项目代码，实现协同开发。在GitHub上实现协同开发一般有以下三种形式。

3.1　添加协作者

项目代码仓库的拥有者可以将协同开发人员直接添加至仓库中，赋予协作者特定的权限。

仓库的拥有者A要想协作者B参与该项目的团队开发，可以通过邀请操作将B添加到该项目协作中，赋予B等同于拥有者的权限。此时。B需要做的是先克隆A的代码仓库到其本地IDE中，再提交代码（commit/push/merge/branch）的权限，就好比B拥有A的repo一样，但是B操作A repo仓库时使用的是B自己的GitHub账号和密码。同样，此时B自己的GitHub账号中并不会出现A的仓库repo，这完全不同于Fork代码。

3.2　关联复制请求方式

对于开源项目间的合作，一般都会使用关联复制请求的合作方式。

假设此时A拥有一个开源仓库，B想参与该开源项目，要先关联复制A开源项目的代码仓库到自己账号的代码仓库中，这样B便拥有了一个A开源项目的镜像项目源码。B可以在本地正常修改自己账号下A的开源项目镜像，B完成自己的开发任务后便可以向A提出代码整合请求。

3.3　组织方式

组织的管理者A可以在创建完毕的team中添加项目源码，并设置一定的权限，添加team成员，实现team成员间的协同开发。

4　结语

在软件开发项目管理过程中，灵活使用版本控制工具，在空间上对整个软件开发项目进行集

中管理，可以有效解决代码冲突、事务并发、文档冗余等问题。同时，在开发过程中使用版本控制工具，能够把控开发进度，有效记录版本的历次变更，并在必要时支持版本回溯。另外，项目管理人员可以使用版本控制工具控制团队成员在开发过程中的操作权限，对负责不同功能点的协同开发人员赋予不同的操作权限，以避免不同分工的协作者误操作带来一系列问题，同时降低人为错误。

在团队协作项目中，使用 GIT 进行版本控制可以保证有记录版本的可回溯性，其本地化版本库的特性及强大的分支能力可以有效把控软件开发项目的质量，提高管理效率。

参考文献

[1] 戴楠，闫明星．用 SVN 实现软件的版本控制 [J]．电脑知识与技术，2009，5（16）：4289-4290，4293.

[2] 宋冬生．Git——版本管理之利器 [J]．程序员，2007（11）：118-119.

[3] 庞双玉．GIT 分布式版本控制实现机制探讨 [J]．信息系统工程，2018（10）：53-54.

[4] 刘悦之．基于 GIT 的分布式版本控制系统的设计与实现 [J]．科技传播，2012，4（22）：197-198.

[5] 张萱．用 SVN 实现软件的版本控制 [J]．电子技术与软件工程，2017（10）：63. PMT

收稿日期：2020-01-08

作者简介：

王真（1994—），女，研究方向：项目管理。