封面

内封

摘要

目录

前言

自从世界上第一台全自动电子计算机“埃尼阿克”诞生以来，计算机技术的发展极其迅速。在移动互联网及大数据的背景下，计算机已经融入到人类生活的方方面面。从衣食住行到工业生产，计算机已经成为人类社会不可或缺的重要组成部分。随着计算机科学与技术的快速发展而来的各种规模的软件工程，常常需要整合人力以及各种软件开发工具，这时版本控制的概念应运而生。

版本控制作为软件工程的一种技巧，需要让项目中的多个开发人员的开发进度保持同步。在软件开发过程中，参与项目的人数多不代表一定可以加快项目的开发速度，因为软件产品的抽象性，人员之间的沟通与管理极有可能占用很多资源，导致整体开发效率下降，而版本控制系统的作用就是协调开发人员的工作，同步不同开发者的进度，尽可能减少版本管理占用的资源。

现代软件开发已经不是“软件=数据结构+算法”的模式，而是要适应各种变化的需求，使软件架构有足够的灵活性，不至于因为需求的变化而推倒重来。现代软件源于互联网的发展，互联网使得软件进入新的时代。互联网深入生产生活的方方面面，因此需要处理一些难以用算法表达的业务逻辑，如银行的金融业务不仅很复杂，很难用算法表示，而且经常调整，导致需求多变。这就需要版本控制适应变化的需求，可以进行同一系统下的分支管理。

一个算法可以从源代码中识别出来，而业务逻辑则很难从代码中看出来。一些企业级软件即使留下源代码，后来者也很难明白其中的业务逻辑。随着老一代程序员的退休，他们也将业务逻辑带走了。以至于后来者不敢轻易重构遗留代码，怕一个误解造成巨大损失。为了解决这样的问题，版本控制系统需要记录下迭代过程中的版本描述，以简化维护人员分析项目的过程。

源代码是开发人员的产品，因此对版本控制系统的安全性及稳定性有较高的要求。软件及硬件上的错误是无法预料的，同时也很难避免，因此对于版本控制系统的文件系统这方面，分布式架构是一个非常好的选择。分布式系统具有较高容错性，将文件操作等较慢的操作分发给多个服务器也有助于提高文件读写效率，采用RAID技术的底层文件系统对文件安全也有很大保障。

目前主流的版本控制系统有github、svn、rcs和cvs等，每一种版本控制系统都有其优点，我将在其中选择有代表性的功能进行实现。在开发这个系统的过程中，我运用本科四年学到的理论知识，以理论结合实际，磨练编程技巧，拓展开发技术，让我在软件开发这一领域有了更深入的理解。

1 需求分析

1.1 需求分析引言

1.1.1 项目背景

本系统的名称为基于ASP.NET的源代码版本控制系统。版本控制是一种记录一个或若干文件内容变化，以便将来查阅特定版本修订情况的系统。软件工程师或网页设计师可能会需要保存某一系统的源码或页面布局文件的所有修订版本，采用版本控制系统是个明智的选择。有了版本控制系统就可以将某个文件回溯到之前的状态，甚至将整个项目都回退到过去某个时间点的状态。用户可以通过系统比较文件的变化细节，查出最后是团队中的哪个成员修改了哪个地方，从而找出导致问题出现的原因，是谁在何时报告了某个功能缺陷等等。使用版本控制系统通常还意味着，就算对整个项目中的文件进行大幅度改动，也照样可以轻松恢复到原先稳定版本的样子，但额外增加的工作量却微乎其微。

版本管理是软件配置管理的基础，它管理并保护开发者的软件资源。本系统作为软件开发过程中的辅助工具，目的在于减少版本控制及管理过程中使用的人力资源，协调进行不同工作的开发人员的工作，同步不同开发者的进度。系统需要尽可能保存每一阶段的工作成果，尤其是源文件，以保证每个阶段性工作成果的安全，这样任何时候都可以方便的找回原来的工作成果；另一方面系统应能够快速检索工作成果，比如很容易找到某个版本的文件，或者最主要的几个阶段性成果，并且能够很容易预览需要的文件。

1.1.2 项目风险

由于开发能力有限以及时间安排上的任务冲突，可能无法实现开题报告所述的全部功能，但可以保证实现基本的版本控制功能。对于使用者，风险主要来自于程序编写过程产生的漏洞，以及系统的初期版本可能不稳定。此外，因为系统的重点在于版本控制，分布式文件系统可能仅有部分功能可以启用。初期版本系统的部分功能在质量及性能上可能无法达到本需求文档的要求，但后续版本可能会有提升。

1.1.3 本系统的目标用户

本系统的目标用户分为普通用户及系统管理员。普通用户是本系统的最终用户，一般为软件开发人员或软件项目管理人员，这一类用户应具有版本控制的基本知识，可以理解计算机的基本操作及运行原理。使用者应尽可能保存软件开发过程中的各个阶段，以及各版本对应的描述性说明及详细开发文档。对于每一个使用本系统的软件项目，本系统的使用频度应对应于项目的开发速度，可以每次修改源代码就使用本系统，也可以在某开发进程达到某阶段后使用本系统。

系统管理员对本系统进行管理，其具有直接操作后台数据库以及配置本系统的权限（如备份、回滚数据库等），并在系统出现问题时对普通用户进行回应。

1.2 功能需求

1.2.1 系统范围

代码仓库：具有受版本控制的所有文件的完整修订历史的共享数据库。

分支：分支是指目录和文件的现有原始树的副本。分支的生命周期是从某事物的副本开始的，并从此副本处移动，生成自己的历史。通常创建分支以尝试新功能，同时不影响具有编译器错误和小问题的开发的主分支。

普通用户可以创建软件项目对应的代码仓库；代码仓库有且只有一个主分支，该分支无法被删除且至少有一个版本；用户可以在代码仓库下创建分支，分支的根可以不是主分支；用户可以查看选中的代码仓库，也可以查看选中的分支及版本；用户可以向代码库中选中的版本上传文件、下载文件或创建文件夹；系统应记录每一次版本迭代的信息，如描述和时间等;用户可以在代码仓库下提交评论和文件，文件经过仓库拥有者审核后可以合并到对应版本中。

代码仓库、分支及版本应该具有项目描述和创建时间等描述性信息。仓库拥有者可以创建、删除分支，可以将当前分支回滚到之前的某一个版本。仓库拥有者也可以删除指定仓库。

普通用户可以创建账号；系统应对已登录用户和未登录用户的权限进行验证。

具有权限的用户可以预览某一版本的文件结构和文件内容，并可以签入或签出选中的文件。如果时间允许，系统应实现不同版本文件之间的差异对比。

分布式文件系统应具有文件操作的基本功能，并且分布式服务器应是可拓展的。

1.2.1 系统体系结构

是