МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Гжельский государственный университет»** (ГГУ)

Колледж ГГУ

Специальность 09.02.07 Информационные системы и программирование

**Реферат**

**По предмету «Моделирование и анализ программного обеспечения»**

**На тему «Система контроля версий ПО»**

ВЫПОЛНИЛ:

Студент группы ИСП-0-17

Филипович А.А.

ПРОВЕРИЛА:

Прокуронова А.Ю.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

п. Электроизолятор

2019 г.

# **Понятие системы контроля версий**

# Система управления версиями (Version Control System или Revision Control System) представляют собой программное обеспечение для облегчения деятельности с быстро меняющейся информацией. Система контроля версий предоставляет возможность хранить несколько вариантов одного и того же документа. При необходимости можно вернуться к старым версиям, можно узнать, кем были сделаны те или иные изменения и т.д.

# Такого рода системы в большинстве своем используются при разработке программного обеспечения, чтобы можно было хранить исходные коды разрабатываемых программ. Система контроля версий позволяет разработчикам хранить прошлые версии файлов из разработки и доставать их оттуда. Она хранит информацию о версии каждого файла (и полную структуру проекта) в коллекции, обычно называемой репозиторием. Но тем не менее данные системы могут использоваться и в других областях знаний, которые включают в себя огромное количество часто изменяющихся электронных документов. Например, они всё чаще применяются в САПР, обычно, в составе систем управления данными об изделии (PDM). Управление версиями используется в инструментах конфигурационного управления.

# Внутри репозитория могут быть несколько параллельных линий разработки, обычно называемых ветвями. Это может быть полезно для хранения стабильной или выпущенной версии ветви, одновременно продолжая работу над рабочей версией. Другой вариант - это открыть выделенную ветвь для работы над экпериментальной возможностью.

# Система контроля версий также позволяет пользователям дать ярлык снимку ветви (часто называемых как тэги) для легкого доставания. Это полезно для обозначения индивидуальных релизов или самых свежих рабочих версий, предназначенных для использования.

# Использование системы контроля версий безусловно обязательно для разработчика, если проект больше нескольких сот строк или если для проекта совместно работают несколько разработчиков. Использование хорошей системы контроля версий определенно лучше, нежели использование узко-направленных специальных методов, которые используют некоторые разработчики для управления различными ревизиями своего кода.

# Большинство систем управления версиями используют централизованную модель, когда имеется единое хранилище документов, управляемое специальным сервером, который и выполняет большую часть функций по управлению версиями. Пользователь, работающий с документами, должен сначала получить нужную ему версию документа из хранилища; обычно создаётся локальная копия документа, т. н. «рабочая копия». Может быть получена последняя версия или любая из предыдущих, которая может быть выбрана по номеру версии или дате создания, иногда и по другим признакам. После того, как в документ внесены нужные изменения, новая версия помещается в хранилище. В отличие от простого сохранения файла, предыдущая версия не стирается, а тоже остаётся в хранилище и может быть оттуда получена в любое время. Сервер может использовать т. н. дельта-компрессию - такой способ хранения документов, при котором сохраняются только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

# Очень часто над одним и тем же проектом трудится несколько человек. Если один из них будет изменять исходный файл, и одновременно с этим другой человек будет выполнять аналогичную операцию, то возможна такая ситуация, что какие-то изменения могут не сохраниться. Системы контроля версий работают с такого рода проблемами и имеют определенный перечень их решения. В большинстве своем эти системы могут автоматически объединять такого рода изменения, которые делают разные члены команды разработчиков. Но стоит отметить, что такого рода объединения чаще всего выполняется для текстовых файлов и с определенным условием: изменения происходили в разных частях файла. Данное ограничение имеет место, поскольку в большинстве своем системы контроля версий направлены на поддержку процесса разработки программных продуктов, а изначальные коды находятся в текстовых файлах. В случае если автоматически выполнить объединение не получилось, то система предлагает исправить ситуацию вручную.

# Зачастую осуществить объединение не возможно ни с помощью системы, ни вручную. Ярким примером является ситуация когда формат файла очень сложный или неизвестен. Отдельные системы контроля версий предоставляют возможность пользователю заблокировать файл в хранилище. Данная операция не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла и обеспечивает, таким образом, исключительный доступ только тому пользователю, который работает с документом.

# Многие системы управления версиями предоставляют ряд других возможностей:

# 1. Позволяют создавать разные варианты одного документа, т. н. ветки, с общей историей изменений до точки ветвления и с разными -- после неё.

# 2. Дают возможность узнать, кто и когда добавил или изменил конкретный набор строк в файле.

# 3. Ведут журнал изменений, в который пользователи могут записывать пояснения о том, что и почему они изменили в данной версии.

# 4. Контролируют права доступа пользователей, разрешая или запрещая чтение или изменение данных, в зависимости от того, кто запрашивает это действие.

# **Виды систем контроля версий**

# Система контроля версий может быть любой формы и размеров, но есть основные положения об их архитектуре. Некоторые системы поддерживают Атомарные Фиксации, которые значат, что состояние всего репозитория меняется полностью. Без Атомарных Фиксаций, каждый файл или часть меняется отдельно и поэтому состояние всего репозитория в любой точке не может быть зафиксировано.

# Большинство обычных VCS (Version Control System ) систем позволяют объединять изменения между ветвями. Это значит, что изменения, зафиксированные в одной ветви, будут зафиксированы в главной линии или в другой ветви с помощью одной автоматической или, по крайней мере, полуавтоматической операцией.

## Subversion

## Subversion стремится быть лучшей альтернативой CVS (см. приложение 1). Она поддерживает большинство соглашений CVS , включая большую часть набора команд, поэтому пользователи CVS быстро чувствуют себя как дома. Subversion предлагает много полезных улучшений по сравнению с CVS: копирование и переименование файлов и директорий, настоящие атомарные фиксации, эффективная обработка бинарных файлов, способность сетевой работы по HTTP (и HTTPS ). Subversion также имеет Win32 клиент и сервер.

## Таким образом, можно выделить следующие возможности Subversion:

## 1. Реализовано большинство возможностей CVS;

## 2. Отслеживается история файлов, директорий и метаданных файлов, в том числе при переименовании и копировании;

## 3. Публикации изменений атомарны;

## 4. Возможность организации доступа к хранилищу Subversion через Apache по протоколу WebDAV/DeltaV;

## 5. Возможность установки автономного сервера Subversion с доступом по собственному протоколу;

## 6. «Дешёвые» операции создания ветвей и меток (требуется небольшое фиксированное количество временных и дисковых ресурсов);

## 7. Многоуровневая архитектура библиотек, изначально рассчитанная на клиент-серверную модель;

## 8. Клиент-серверный протокол пересылает по сети только разницу между объектами, когда это возможно;

## 9. Затраты ресурсов пропорциональны размеру изменений, а не размеру данных, которые затронуты изменениями;

## 10. Два возможных внутренних формата репозитория: база данных или простой файл;

## 11. Версионированные символьные ссылки (только в рабочих копиях под UNIX-системами);

## 12. Одинаково эффективная работа и с текстовыми, и с двоичными файлами;

## 13. Вывод клиента командной строки одинаково удобен и для чтения, и для разбора программами;

## 14. Частичная локализация сообщений (используются настройки локали);

## 15. Библиотеки для языков PHP, Python, Perl, Java;

## 16. Возможность зеркалирования репозитория.

**Git**

### Git является распределённой системой контроля версий. В нем файлы могут находиться в одном из трёх состояний: зафиксированном, изменённом и подготовленном. "Зафиксированный" значит, что файл уже сохранён в вашей локальной базе. К изменённым относятся файлы, которые поменялись, но ещё не были зафиксированы. Подготовленные файлы - это изменённые файлы, отмеченные для включения в следующий коммит.

### Таким образом, в проектах, использующих Git, есть три части: каталог Git'а (Git directory), рабочий каталог (working directory) и область подготовленных файлов (staging area).

### Каталог Git'а -- это место, где Git хранит метаданные и базу данных объектов вашего проекта. Это наиболее важная часть Git'а, и именно она копируется, когда вы клонируете репозиторий с другого компьютера.

### Рабочий каталог -- это извлечённая из базы копия определённой версии проекта. Эти файлы достаются из сжатой базы данных в каталоге Git'а и помещаются на диск для того, чтобы вы их просматривали и редактировали.

### Область подготовленных файлов -- это обычный файл, обычно хранящийся в каталоге Git'а, который содержит информацию о том, что должно войти в следующий коммит. Иногда его называют индексом (index), но в последнее время становится стандартом называть его областью подготовленных файлов (staging area).