0) Общее описание и история развития систем контроля версий.

Изначально в разработке ПО и некоторых смежных областях для дифференциации версий информации использовали самый простой метод, при котором создавалась папка с числом, соответствующим текущей версии документа, например, diplom05\_03\_2015, или diplom\_v2.1.0.115. Это всё ещё очень популярный метод версификации, потому что он очень простой, не требует дополнительного ПО на любой современной ОС и полностью интуитивно понятный.

Тем ни менее, недостатки такого метода достаточно серьёзны: во-первых, человеку свойственны ошибки, поэтому иногда можно перепутать текущую версию документа, работать в директории, отличающейся от актуальной, а также необходимость файлового хранилища большего размера для всех версий файлов.

Чтобы исправить эти ошибки, стали появляться системы контроля версий, или VCS (Version Control System) – такие системы, которые позволяют хранить изменения в файле, множестве файлов или папок таким образом, чтобы можно было в любой момент времени восстановить информацию до некого состояния файла в прошлом, и, кроме этого, сделать это с минимально возможными затратами.

Лучше всего VCS работает с текстовыми файлами, так как в таком случае проще всего найти точное место, где он был изменён. Тем ни менее, хранить можно любые типы файлов; единственная проблема будет заключается в отсутствии возможности выбрать (или применить, просмотреть) какие-либо отдельные изменения, не меняя весь файл целиком.

Самой первой системой контроля версий стала SCCS – (Source Code Control System). Она использовала технику чередующихся изменений, которая теперь, в современных СКВ сменилась на версификацию и слияние.

[0] <https://ru.wikipedia.org/wiki/Source_Code_Control_System> (?)

После этого появились RCS (Revision Control System), как более развитая альтернатива SCSS. В связи с отсутствием ПО, которое предоставляло бы схожий функционал, RCS быстро завоевали популярность не только у разработчиков, но и некоторых других категорий пользователей. К слову, RCS, созданная в 1982 году, до сих пор поставляется в стандартной сборке Developer Tools с Mac OS X.

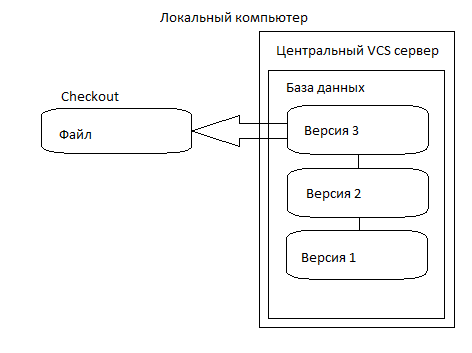


Рисунок 0 – Диаграмма взаимоотношений в RCS.

1) Обзор централизованных и децентрализованных систем контроля версий.

Тем ни менее, несмотря на успешность и тот факт, что RCS по-прежнему используется в реальных проектах (TWiki), в этой системе существует масса проблем, которые и помешали дальнейшему распространению такой системы. Так, например, RCS взаимодействует только с выбранными файлами: нет возможности работать над все проектом целиком. Кроме того, несмотря на наличие функции разветвления (branching), синтаксис поведения очень тяжёл для понимания. Поэтому многие команды используют встроенный механизм блокировки и работают в одной ветке head. На практике это означает, что только один пользователь может менять какой-либо файл в данный момент (lock-unlock механизм); пока он не закончит свою работу над ним, никто не сможет изменить данный файл.

[1] <http://en.wikipedia.org/wiki/Revision_Control_System#Mode_of_operation>

На самом деле, с приходом более совершенных IDE и языков программирования, вопрос взаимодействия с коллегами является ключевым для многих разработчиков. В основном чтобы решить эту проблему, начали появляться централизованные системы контроля версий (ЦСВ или CVS – Centralized Version Control systems). Примерами таких систем являются:

* CVS;
* Subversion;
* Perforce.

В них используется следующий метод сохранения информации: имеется один сервер, который хранит все версии файлов, а также некоторое количество клиентов, на которые забирается информация с сервера. Диаграмма таких взаимоотношений представлена на рисунке 1:

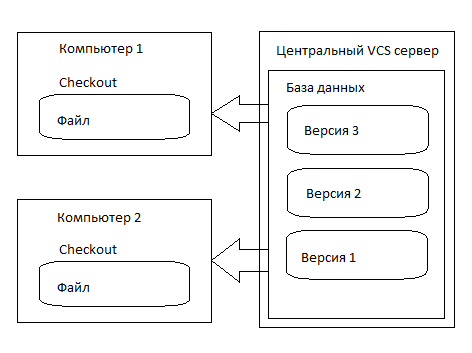


Рисунок 1 – Диаграмма взаимоотношений в VCS

Такой подход имеет много преимуществ, особенно относительно RCS. Например, каждый разработчик в любой момент времени может знать, что делают все остальные; кроме этого, администраторы систем, имея точно такую же возможность, могут эффективнее управлять своими кадрами и всей системой в целом.

Но, несмотря на решение достаточно большого спектра проблем, у таких систем есть свои недостатки. Первый и наиболее существенный – единая точка отказа. Если сервер по каким-то причинам выходит из строя, то все работают только со своими локальными копиями, не имея возможности смотреть изменения других разработчиков.

2) Типовые операции и порядок работы с системой контроля версий.  
Если не вдаваться в подробности реализации и работы с системами контроля версий, то общий порядок работы примерно одинаков для любой VCS.

Первое, что делает любой разработчик, начиная работать с проектом – создание рабочей копии проекта. В Subversion для этих целей служит команда checkout, для Git – clone. После выполнения этой команды на рабочей машине разработчика появляется директория, в которой находятся непосредственно сам проект. После этого можно непосредственно приступать к разработке.

Стоит отметить, что, в зависимости от реализации, может установиться не одна папка, а две: одна с рабочей копией, а вторая – полный её дубль, позволяющий в любой момент времени локально определить, какие файлы были изменены.

Далее разработчик будет модифицировать рабочую копию директории, чтобы в дальнейшем зафиксировать изменения, отправляя их на сервер.

В случае, если число разработчиков больше, чем 1, то зачастую будут встречаться конфликтные ситуации, когда разработчики одновременно вносят правки в одну и ту же копию документа.

3) Разработка концепции лабораторного практикума.