0) Общее описание и история развития систем контроля версий.

Изначально в разработке ПО и некоторых смежных областях 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 версионирования, потому что он очень простой, не требует дополнительного ПО на любой современной ОС и полностью интуитивно понятный.

Тем ни менее, проблемы такого метода достаточно серьёзны: во-первых, человеку свойственны ошибки, поэтому иногда можно перепутать текущую версию документа, работать в директории, отличающейся от актуальной и тому подобные.

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.

Лучше всего VCS работает с текстовыми файлами, так как различия между ними лучше всего заметны. Тем ни менее, хранить можно любые типы файлов; единственная проблема в таком случае будет заключаться в отсутствии возможности выбрать (или применить, просмотреть) какие-либо отдельные изменения.

Самой первой системой контроля версий стала RCS (Revision Control System), которая до сих пор поставляется в стандартной сборке Developer Tools с Mac OS X.

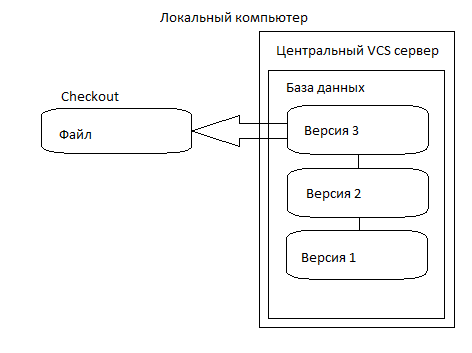


Рисунок 0 – Диаграмма взаимоотношений в RCS.

1) Обзор централизованных и децентрализованных систем контроля версий.

Вторая большая проблема для разработчиков ­– вопрос взаимодействия с коллегами. Чтобы решить эти проблемы, была создана централизованная система 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:

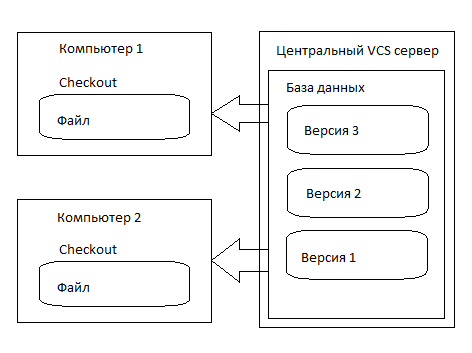


Рисунок 1 – Диаграмма взаимоотношений в VCS

Такой подход имеет много преимуществ, особенно относительно VCS. 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

(пункт 2 можно начинать делать из википедии, в русской статье есть подходящий раздел).

3) Разработка концепции лабораторного практикума.