Контрольные вопросы:

1) **Централизованные системы контроля версий**

Следующая серьёзная проблема, с которой сталкиваются люди, — это необходимость взаимодействовать с другими разработчиками. Для того, чтобы разобраться с ней, были разработаны централизованные системы контроля версий (ЦСКВ). Такие системы, как CVS, Subversion и Perforce, используют единственный сервер, содержащий все версии файлов, и некоторое количество клиентов, которые получают файлы из этого централизованного хранилища. Применение ЦСКВ являлось стандартом на протяжении многих лет.

Такой подход имеет множество преимуществ, особенно перед локальными СКВ. Например, все разработчики проекта в определённой степени знают, чем занимается каждый из них. Администраторы имеют полный контроль над тем, кто и что может делать, и гораздо проще администрировать ЦСКВ, чем оперировать локальными базами данных на каждом клиенте.

Несмотря на это, данный подход тоже имеет серьёзные минусы. Самый очевидный минус — это единая точка отказа, представленная централизованным сервером. Если этот сервер выйдет из строя на час, то в течение этого времени никто не сможет использовать контроль версий для сохранения изменений, над которыми работает, а также никто не сможет обмениваться этими изменениями с другими разработчиками. Если жёсткий диск, на котором хранится центральная БД, повреждён, а своевременные бэкапы отсутствуют, вы потеряете всё — всю историю проекта, не считая единичных снимков репозитория, которые сохранились на локальных машинах разработчиков. Локальные СКВ страдают от той же самой проблемы: когда вся история проекта хранится в одном месте, вы рискуете потерять всё.

Распределённые системы контроля версий

Здесь в игру вступают распределённые системы контроля версий (РСКВ). В РСКВ (таких как Git, Mercurial, Bazaar или Darcs) клиенты не просто скачивают снимок всех файлов (состояние файлов на определённый момент времени) — они полностью копируют репозиторий. В этом случае, если один из серверов, через который разработчики обменивались данными, умрёт, любой клиентский репозиторий может быть скопирован на другой сервер для продолжения работы. Каждая копия репозитория является полным бэкапом всех данных.

Более того, многие РСКВ могут одновременно взаимодействовать с несколькими удалёнными репозиториями, благодаря этому вы можете работать с различными группами людей, применяя различные подходы единовременно в рамках одного проекта. Это позволяет применять сразу несколько подходов в разработке, например, иерархические модели, что совершенно невозможно в централизованных системах.

2) Локальные системы контроля версий обычно хранят на компьютере список изменений, внесенных в файлы. Основываясь на этих данных, система контроля версий воссоздает нужную версию файла (актуальную на определенный момент времени).

Централизованные системы контроля версий предполагают сохранение версий проектов на общий сервер, с которого потом получают нужные версии клиенты.

Распределенные системы контроля версий позволяют хранить репозиторий (его копию) у каждого разработчика, работающего с данной системой. При этом можно условно выделить центральный репозиторий в который будут отправляться изменения из локальных и, с ним же эти локальные репозитории будут синхронизироваться. При работе с такой системой, пользователи периодически синхронизируют свои локальные репозитории с центральным и работают непосредственно со своей локальной копией.

3) По умолчанию TortoiseSVN использует модель копирование-изменение-слияние, и в большинстве случаев это всё, что вам нужно.

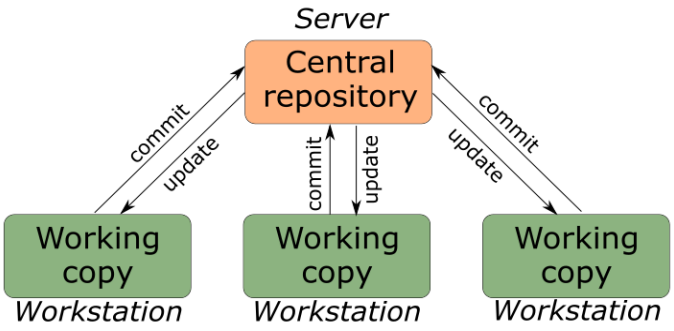
В этой модели клиент каждого пользователя считывает из хранилища проект и создаёт персональную рабочую копию - локальное отражение файлов и каталогов хранилища. После этого пользователи работают, одновременно изменяя свои личные копии. В конце концов, личные копии сливаются в новую, финальную версию. Обычно система управления версиями выполняет слияние автоматически, но, естественно, в общем случае необходимо присутствие человека.

4) «Обновить» - получить последнюю версию из репозитория.

«Фиксировать» - поместить изменения в репозиторий.

«Слить» - добавить изменения в текущую ветку.

5) Работа с git (РСКВ) заключается в следующем: разработчик копирует центральный репозиторий (копия полностью совпадает с оригиналом), создавая отдельные ветви для работы. После внесенных изменений файлы отправляются на сервер.



commit - зафиксировать в коммите проиндексированные изменения (закоммитить).

checkout - для копирования файлов из истории или сцены в рабочую директорию. Также она может использоваться для переключения между ветками.

branch - показать список веток.

reset – отмена индексации файла перед коммитом

add – добавить файлы в коммит

status – проверка файлов на индексацию коммитом (добавлены или нет)

diff – просмотр различий между ветвями

clone – клонирование удаленного репозитория

pull – извлечение и загрузка содержимого из удаленного репозитория и немедленного обновления локального репозитория этим содержимым

push – отправить изменения на сервер