**Системы контроля версий - назначение, примеры решений.**

Система контроля версий — это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии.

* На основе файловой системы

Данный подход является устаревшим. Подразумевалось, что разработчики будут пользоваться центральным сервером с общим доступом к файлам, как только один разработчик решал как-то изменить файл. Другие разработчики не могли получить к нему доступ и ждали пока тот закончит. Пример - NFS

* Централизованные

Репозиторий с данными храниться на сервере, все разработчики имеют к нему удаленный доступ по специальным протоколам. Пример – Subversion

* Распределенные

Репозиторий с данными храниться на сервере, все разработчики клонируют его к себе на локальный репозиторий, работают над ним, после проверок загружают обратно на главный сервер. Пример Git

**Ревизии и ветки.**

Репозиторий – набор файлов, организованных определенной иерархией, для удобной работы с проектом. В SVN каждая фиксация изменений (svn commit) пришедшая в репозиторий повышает версию (ревизию) этого репозитория на 1. Т.е. Ревизия репозитория – целое число, уникально характеризующее набор файлов находящиеся там на данный момент времени.

SVN

* Trunk – Основная ветвь разработки
* Branches – Отдельные модификации продукта, которые могут серьёзно отличатся от основной. После проведения всех изменений ветки сливают с основной веткой.
* Tag - Хранит помеченные версии продукта. Чаще всего используется для устранения дефектов.

GIT

* Master – В данной ветке находятся версии продукта, готовые к поставке
* Develop – Основная разработка
* Feature – Для реализации функциональных требованию создаются данные ветки, в каждой из которых реализуется конкретное, а после сливаются с Develop
* Release – Как только накопилось большое число изменений в Develop, данная версия переносится в Release, где исправляются все дефекты.
* Hotfix – предназначена для исправления критических ошибок в версии пользователя.

**Основные операции над данными в системах контроля версий. Основные команды svn и git.**

**SVN**

* **Извлечение рабочей копии**
  + **svn co svn+ssh://<host or IP>/repo\\trunk [targetDir]**
* **Обновление репозитория**
  + **svn up**
* **Работа с файлами**
  + **svn add src/main/java/Application.java**
  + **svn delete src/main/java/Application.java**
  + **svn copy src/main/java/Service1.java src/main/java/Service2.java**
  + **svn move src/main/java/Service1.java src/main/java/Service2.java**
* **Проверка текущего состояния**
  + **svn status**
  + **svn status --show-updates**
  + **svn diff**
* **Фиксация изменений и отправка на сервер**
  + **svn commit -m "PROJ-0001 Description"**
* **Разрешение конфликта**
  + **svn resolved src/main/java/Application.java**
* **Откат изменений**
  + svn merge -c -100500 .

**GIT**

* git status - показывает статус текущих состояний файлов в файловой системе и информацию о ветви, в которой производится редактирование.
* git log показывает журнал коммитов, а опция --graph выводит в графическом виде ветви, в которых производились данные изменение.
* git diff - покажет все изменения, которые были сделаны относительно последних зафиксированных изменений.
* git reset --hard HEAD сбросить все изменения, которые были сделаны в текущем локальном репозитории.
* git branch - показывает ветви.
* git checkout - переключает разработчика между ветвями
* git merge - объединяет несколько ветвей в текущую ветвь
* git commit - фиксирует изменения в текущей ветке. Опция -m задает сообщение коммита, которое будет показываться пользователю
* git add - добавляет измененные файлы к последующему коммиту, помещая их в Stage Area.

**Виды конфликтов и способы их решения.**