Система управления версиями Subversion (SVN)

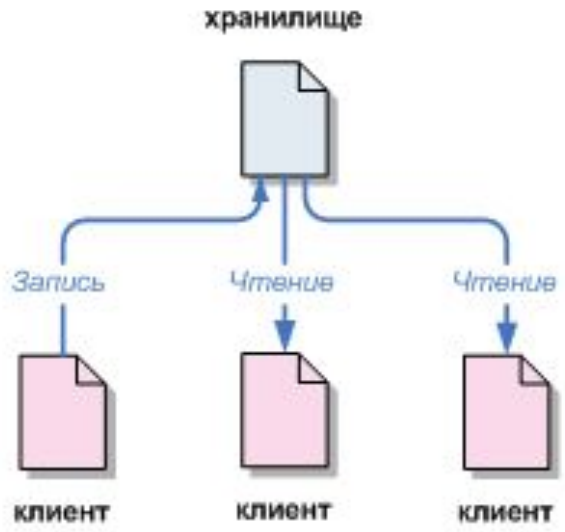
По материалам электронной книги
Управление версиями в Subversion

http://svnbook.red-bean.com/

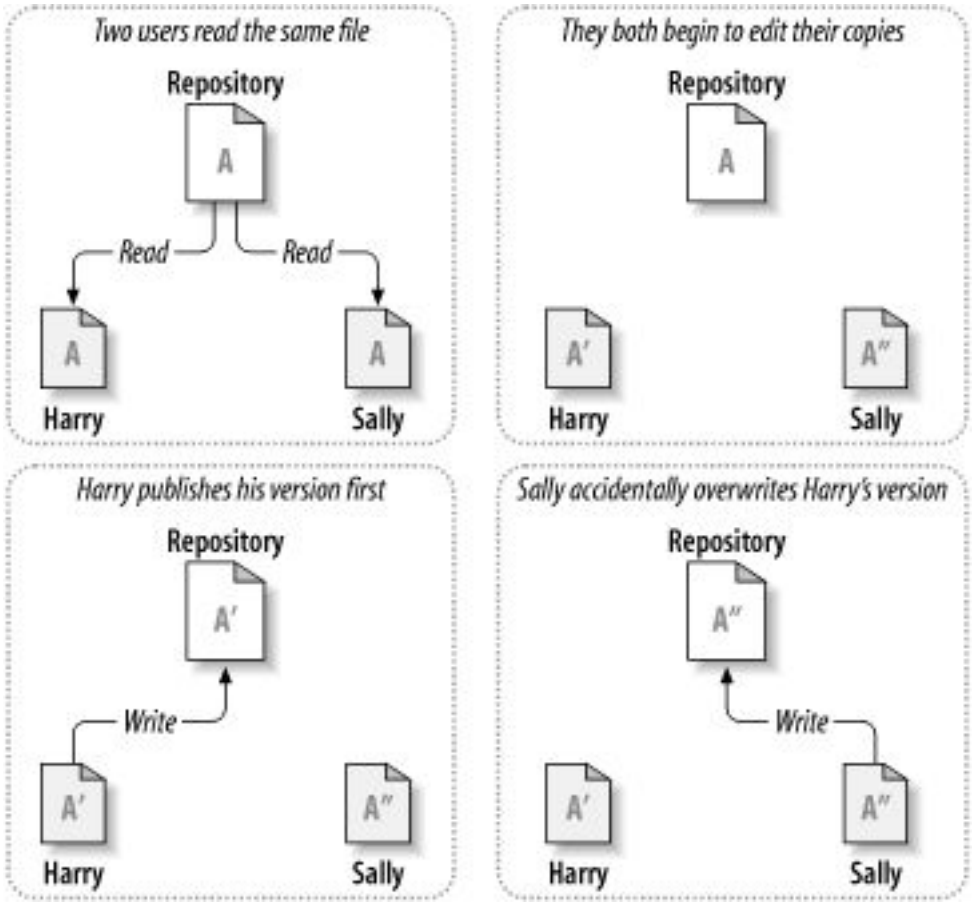
Иртегов Д.В, НГУ-Parallels

Зачем нужны системы управления версиями

- Много пользователей работают с разделяемым ресурсом



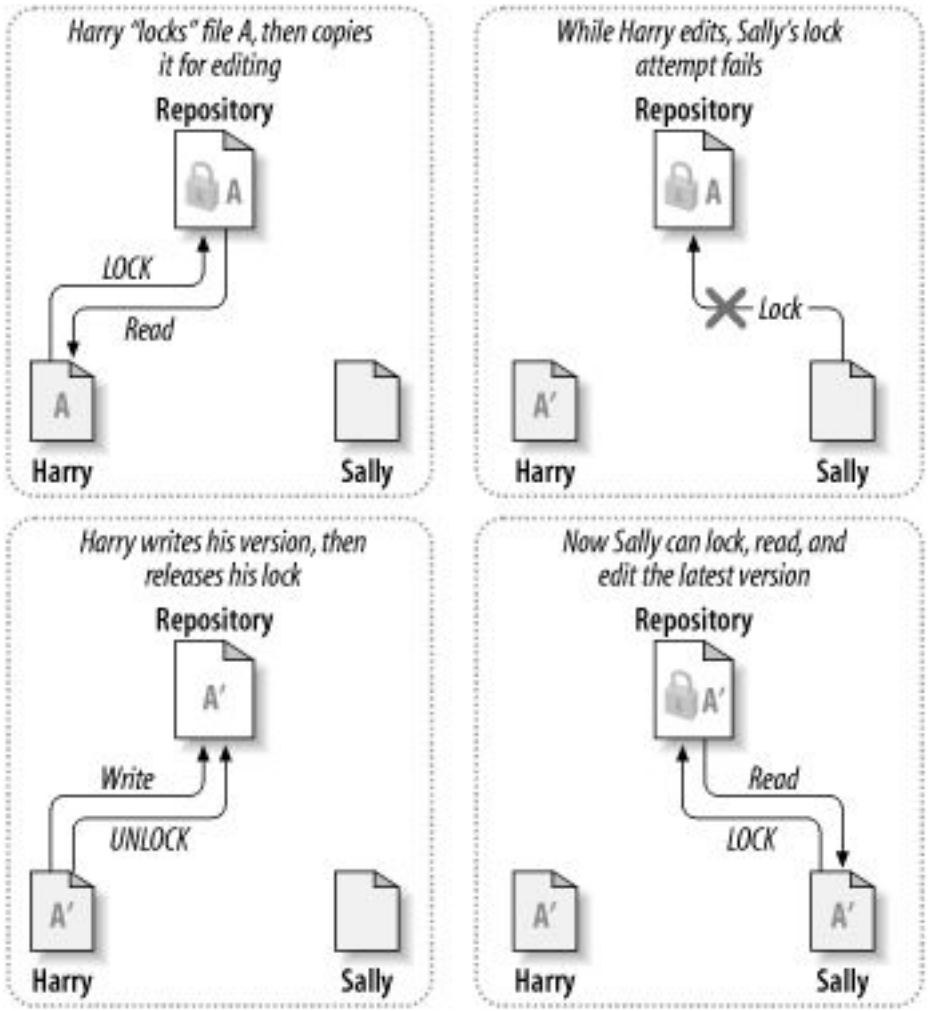
Конфликты


Two users read the same file
Repository
A
Read
Read
A
A
Harry
Sally
They both begin to edit their copies
Repository
A
A'
A''
Harry
Sally
Harry publishes his version first
Repository
A'
Write
A'
A''
Harry
Sally
Sally accidentally overwrites Harry's version
Repository
A''
Write
A'
A''
Harry
Sally

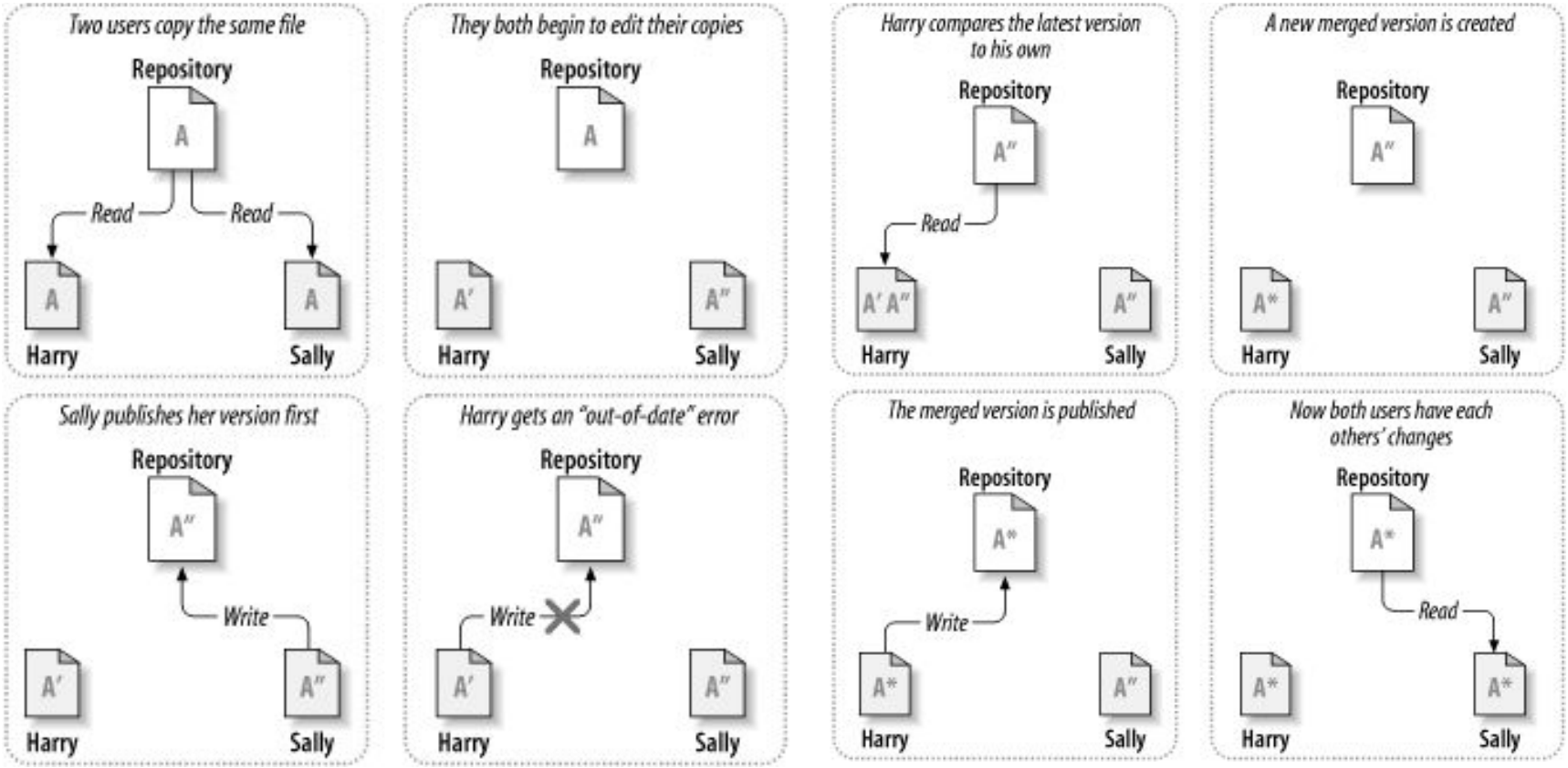
Варианты решения

- Блокирование-Изменение-Разблокирование
- Копирование-Изменение-Слияние

Блокирование-изменение



Копирование-слияние



Использование Subversion

```
$ svn checkout http://svn.example.com/repos/calc
A calc/Makefile
A calc/integer.c
A calc/button.c
Checked out revision 56.
$ ls -A calc
Makefile integer.c button.c .svn/
```

Формат SVN URL

Схема	Метод доступа
`file:///`	прямой доступ к хранилищу (на локальном диске)
`http://`	доступ через протокол WebDAV (если Subversion-сервер работает через Apache)
`https://`	то же, что и `http://`, но с SSL-шифрованием
`svn://`	доступ через собственный протокол к серверу `svnserve`
`svn+ssh://`	то же, что и `svn://`, но через SSH-соединение

Основные операции SVN

- svn checkout
  (скачивание ветки репозитория)
- svn commit
  (передача измененных файлов)
- svn update
  (обновление локальной рабочей области)
- Если вы хотите скопировать или переместить элемент в рабочей копии, вы должны использовать команду **svn copy** или **svn move** вместо аналогичных команд операционной системы.

Состояния локального файла

- Не изменялся и не устарел
  - Файл не изменялся в рабочем каталоге, а в хранилище также не фиксировались изменения этого файла со времени создания его рабочей правки. Команды **svn commit** и **svn update** никаких операций делать не будут.
- Изменялся локально и не устарел
  - Файл был изменен в рабочей копии, но в хранилище не фиксировались изменения этого файла последнего обновления рабочей копии. Есть локальные изменения, которые не были зафиксированы в хранилище, поэтому **svn commit** выполнит фиксацию ваших изменений, а **svn update** не сделает ничего.
- Не изменялся и устарел
  - В рабочем каталоге файл не изменялся, но был изменен в хранилище. Необходимо выполнить обновление файла для того, чтобы он соответствовал текущей опубликованной правке. Команда **svn commit** не сделает ничего, а **svn update** обновит вашу рабочую копию файла в соответствии с последними изменениями.
- Изменялся локально и устарел
  - Файл был изменен как в рабочем каталоге, так и в хранилище. **svn commit** потерпит неудачу, выдав ошибку «out-of-date». Файл необходимо сначала обновить; **svn update** попытается объединить локальные изменения с опубликованными. Если Subversion не сможет выполнить объединение самостоятельно, она предложит пользователю разрешить конфликт вручную.

Нумерация правок (revision)

В отличие от большинства систем управления версиями, номера правок в Subversion относятся *ко всем*, а не только к отдельно взятым файлам. Каждый номер правки соответствует целому дереву, отдельному состоянию хранилища после зафиксированного изменения.

Иначе говоря, правка N представляет состояние файловой системы хранилища после выполнения N-ой фиксации.

Когда пользователи Subversion говорят о «правке 5 foo.c», на самом деле речь идет о «foo.c входящем в правку 5».

Заметьте, что правки N и M файла *не обязательно* будут отличаться!

Многие другие системы управления версиями используют пофайловую нумерацию (номер версии файла увеличивается только после изменения этого файла)

Типичный рабочий цикл

- Обновление рабочей копии
  - **svn update**
- Внесение изменений
  - **svn add**
  - **svn delete**
  - **svn copy**
  - **svn move**
- Анализ изменений
  - **svn status**
  - **svn diff**
  - **svn revert**
- Слияние изменений, выполненных другими, с вашей рабочей копией
  - **svn update**
  - **svn resolved**
- Фиксация изменений
  - **svn commit**

Другие полезные команды

- **svn log**
  - Показывает вам развернутую информацию: лог-сообщения, присоединенные к правкам, с указанием даты изменений и их авторов, а также изменения путей к файлам в каждой правке.
- **svn diff**
  - Показывает подробности того, как изменился файл с течением времени.
- **svn cat**
  - Эта команда используется для получения отдельного файла в том виде, в каком он был в конкретной ревизии и вывода его на экран.
- **svn list**
  - Показывает список файлов в каталоге для любой указанной правки

Доступ к старым версиям

- $ svn checkout --revision 1729
  # Checks out a new working copy at r1729
- $ svn update --revision 1729
  # Updates an existing working copy to r1729

Имена ревизий

- HEAD
  - Последняя (или «самая новая») правка хранилища
- BASE
  - Номер правки элемента в рабочей копии. Если элемент редактировался, то «BASE версия» соответствует тому, как выглядел этот элемент до внесения локальных изменений.
- COMMITTED
  - Правка, в которой элемент последний раз изменялся (предшествующая либо равная BASE).
- PREV
  - Правка, непосредственно *предшествующая* той правке, в которой элемент был последний раз изменен. (То есть, фактически, COMMITTED - 1.)

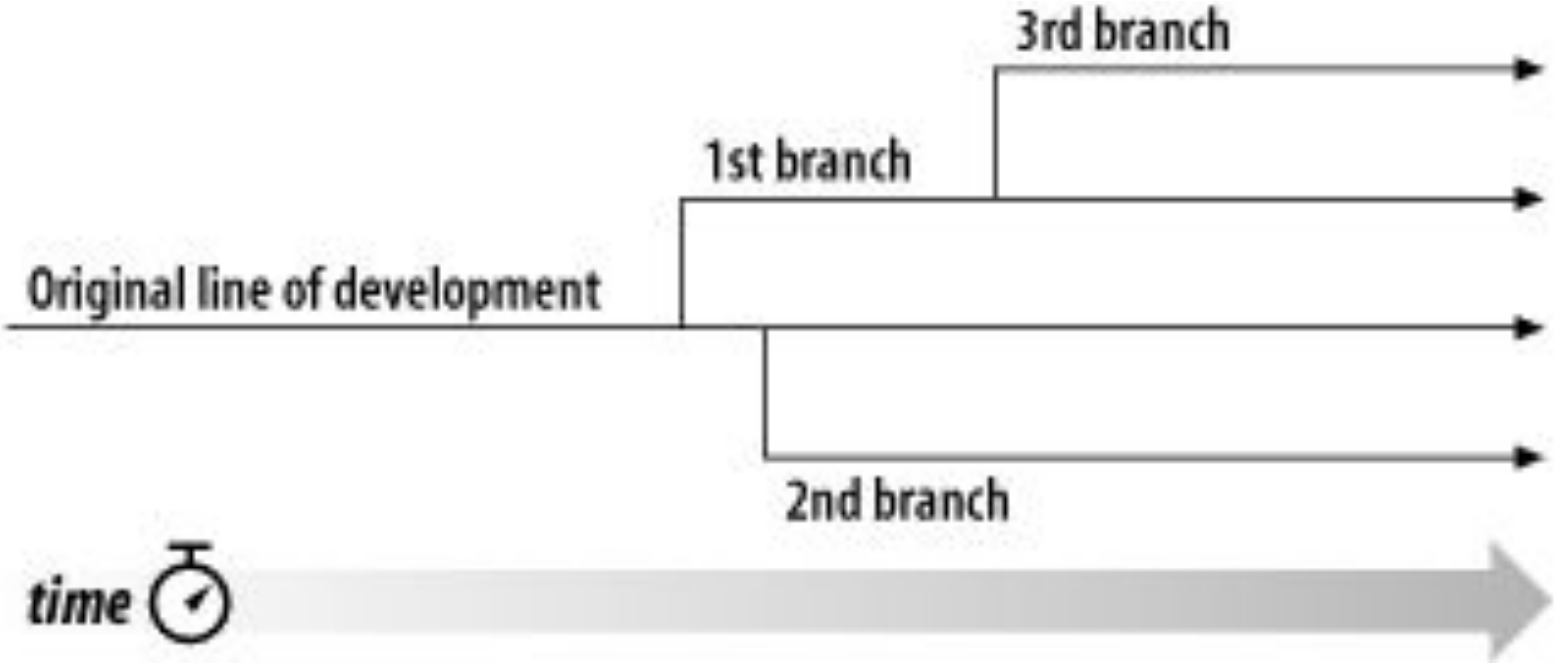
Примеры

- $ svn diff --revision PREV:COMMITTED foo.c
  # показать последнее изменение, зафиксированное для foo.c
- $ svn log --revision HEAD
  # показать лог-сообщение для последней фиксации в хранилище
- $ svn diff --revision HEAD
  # сравнить ваш рабочий файл (с учетом локальных изменений) # с последней правкой в хранилище
- $ svn diff --revision BASE:HEAD foo.c
  # сравнить ваш «исходный» foo.c (без учета локальных
  # изменений) с последней версией в хранилище
- $ svn log --revision BASE:HEAD
  # показать все логи фиксаций со времени вашего последнего обновления
- $ svn update --revision PREV foo.c
  # отменить последние изменения в foo.c, понизив рабочую правку foo.c
- $ svn diff -r BASE:14 foo.c
  # сравнить неизмененную версию foo.c и версию foo.c в правке 14

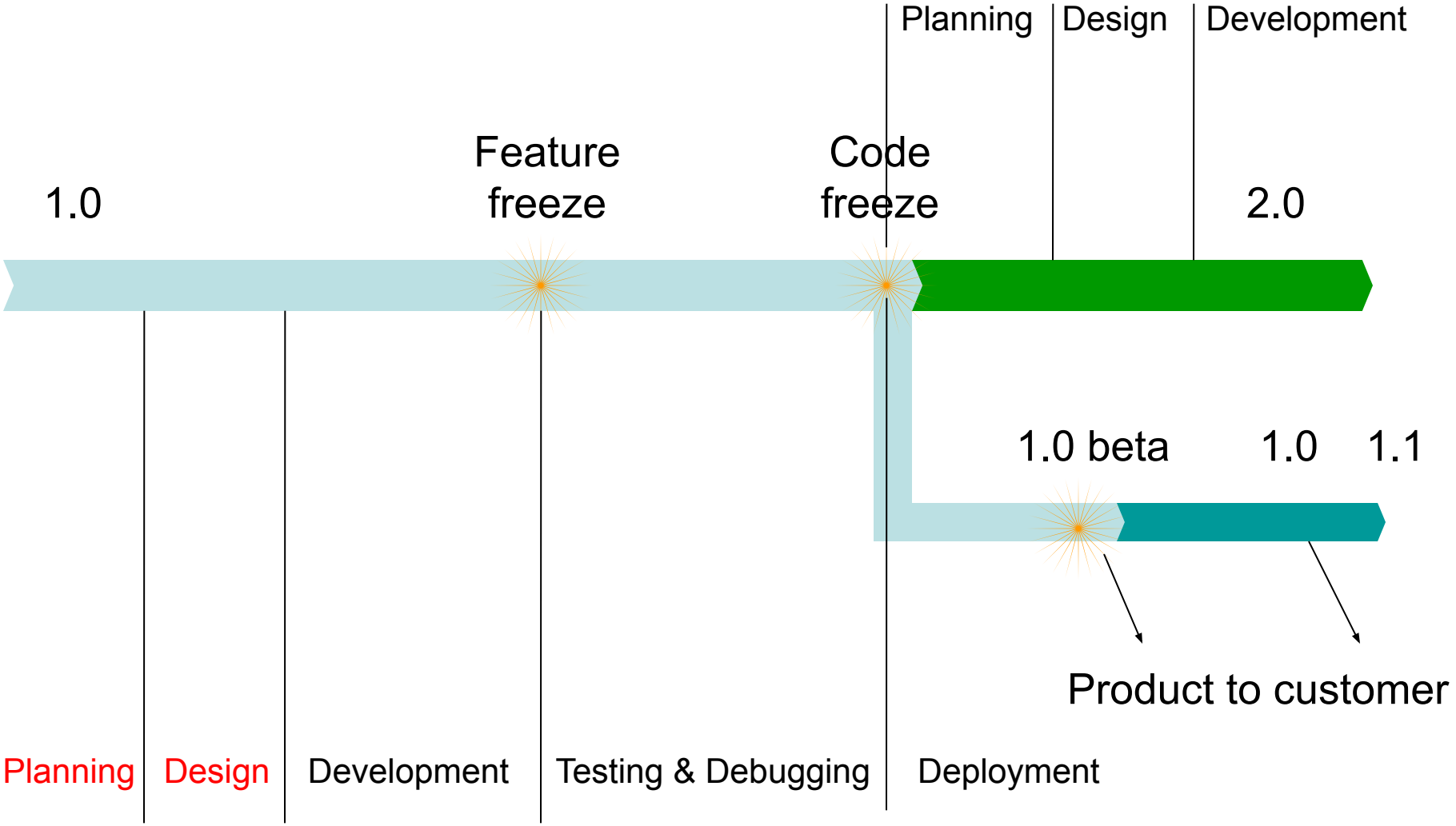
Даты в качестве ревизий

- $ svn checkout -r {2006-02-17}
- $ svn checkout -r {15:30}
- $ svn checkout -r {15:30:00.200000}
- $ svn checkout -r {"2006-02-17 15:30"}
- $ svn checkout -r {"2006-02-17 15:30 +0230"}
- Когда вы указываете дату, Subversion находит в хранилище наиболее близкую правку.

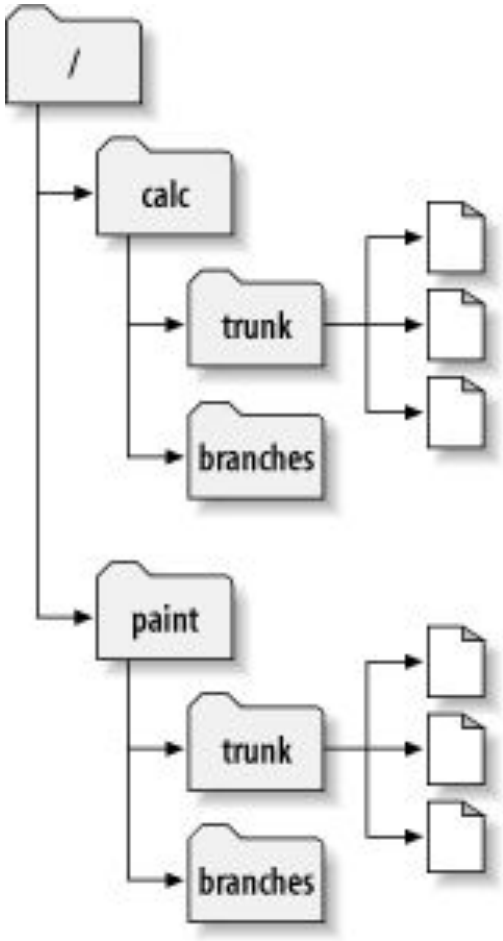
Ветви (branch) и теги (tag)



Зачем нужны ветви?



Структура хранилища



Создание ветви

```
$ svn copy http://svn.example.com/repos/calc/trunk \
   http://svn.example.com/repos/calc/branches/my-calc-branch \
   -m "Creating a private branch of /calc/trunk."

Committed revision 341.
```

Или

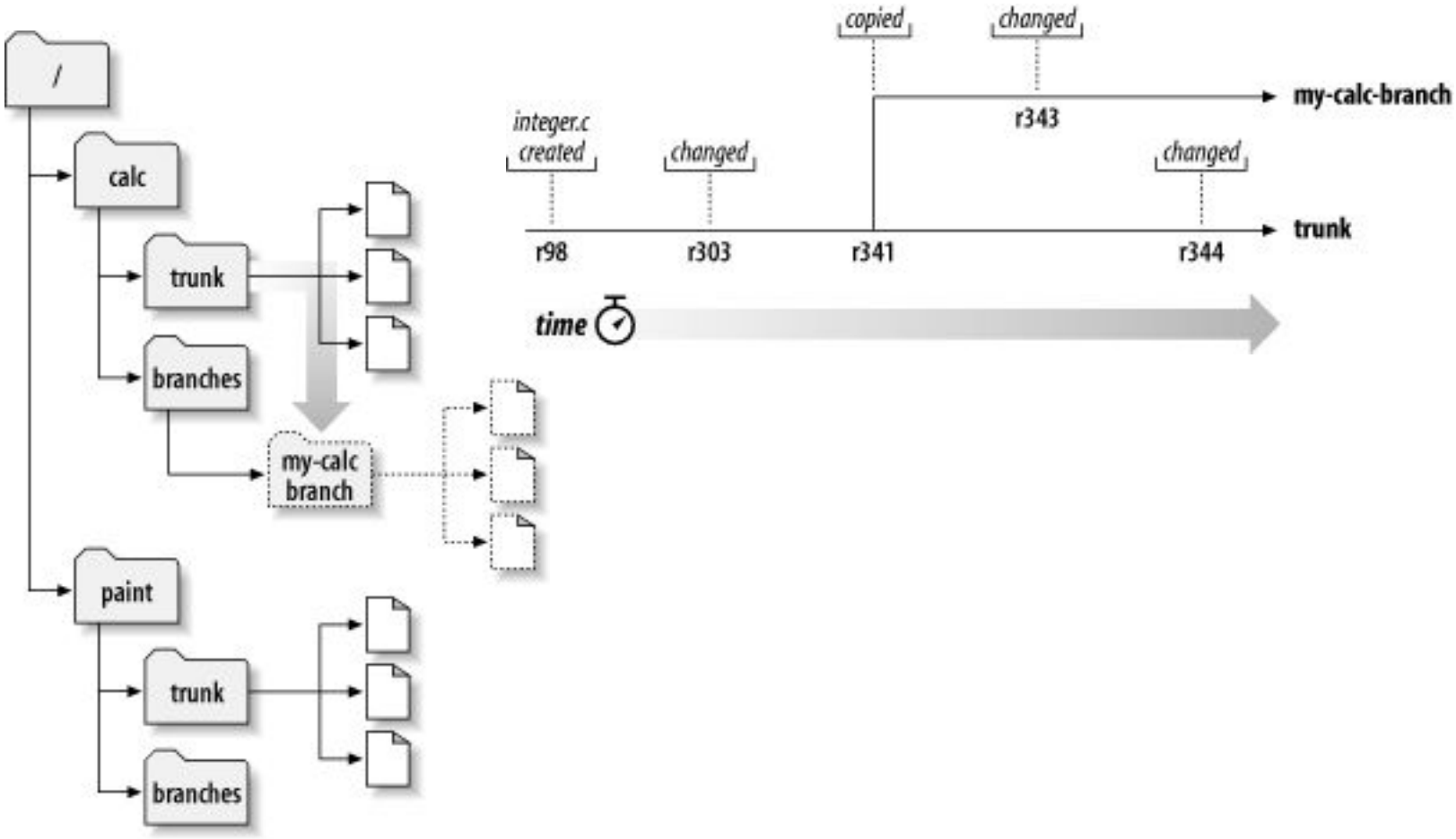
```
$ svn checkout http://svn.example.com/repos/calc bigwc
   A bigwc/trunk/
   A bigwc/trunk/Makefile
   A bigwc/trunk/integer.c
   A bigwc/trunk/button.c
   A bigwc/branches/
Checked out revision 340.

$ svn copy trunk branches/my-calc-branch

$ svn commit -m "Creating a private branch of /calc/trunk."
Adding branches/my-calc-branch

Committed revision 341.
```

Хранилище с веткой



Работа с веткой

- Те же операции, что и со стволовой (trunk) версией
- **svn merge**
  - Строит список различий (diff) между ветвями и применяет их к локальной копии
- Команда принимает три аргумента:
  - Начальное дерево хранилища (как правило, называемое *левой частью* при сравнении),
  - Конечное дерево хранилища (как правило называемое *правой частью* при сравнении),
  - Рабочую копию, к которой отличия применяются в виде локальных изменений (как правило, называемую *целью* слияния).
- **svn switch**
  - трансформирует существующую рабочую копию в другую ветку

Метки (tags)

- Создаются так же, как ветви

```
$ svn copy http://svn.example.com/repos/calc/trunk \
   http://svn.example.com/repos/calc/tags/release-1.0 \
   -m "Tagging the 1.0 release of the 'calc' project."

Committed revision 351.
```

- В отличие от ветвей, в метки [нормальные люди] не коммитят
- Например, тег – это версия, переданная на тестирование
- Если надо, из тега можно потом сделать ветку