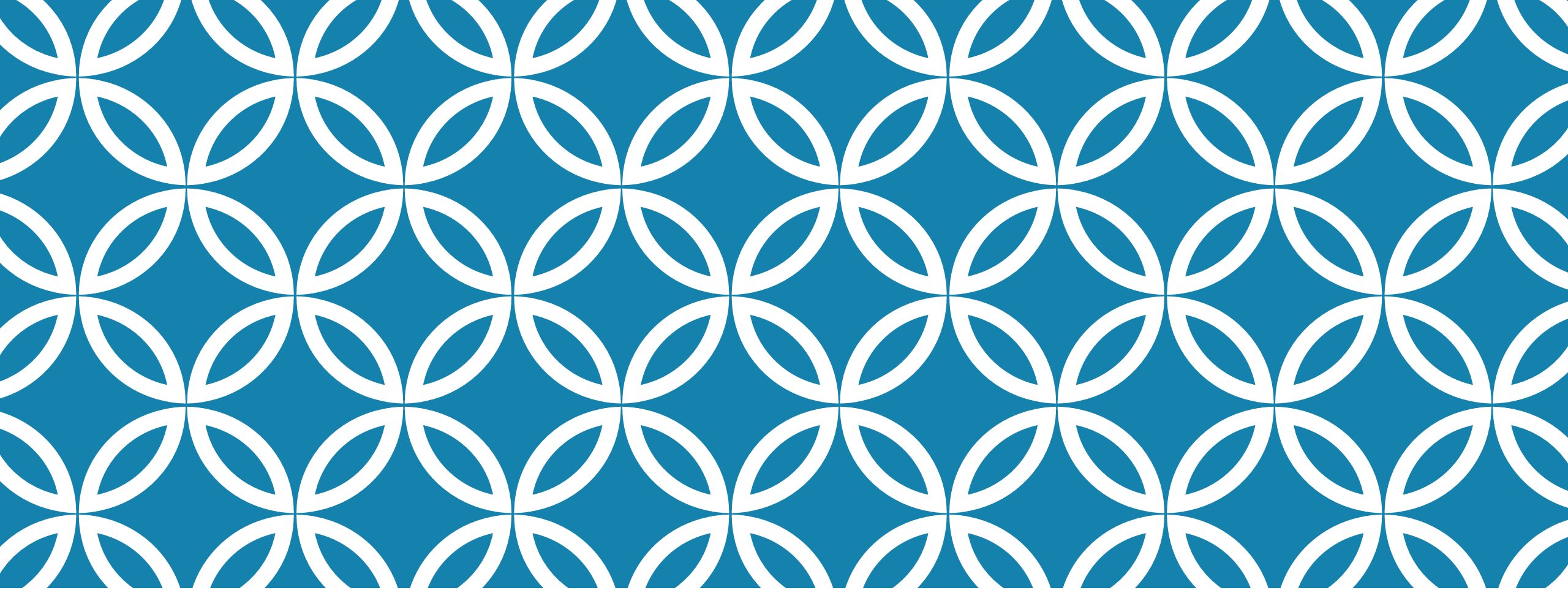


ЛЕКЦИЯ 2.

Subversion. Основные операции.

SUBVERSION

- Централизованная СКВ
- СКВ с открытым исходным кодом
- Лицензия Apache 2.0 (код поставляетмя «как есть» и может использоваться как в открытом, так и в коммерческом ПО)

ИСТОРИЯ СОЗДАНИЯ

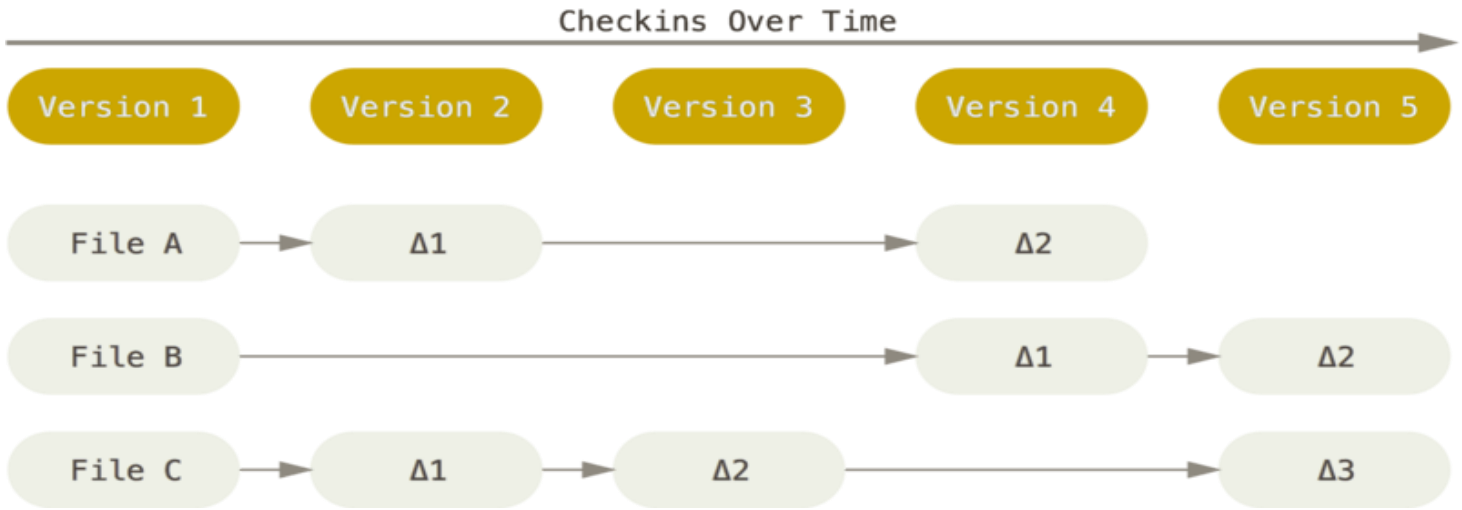
- 2000 годы – CollabNet, Inc решила создать преемника CVS, устранив её недостатки
- Под руководством Карла Фогеля и Бена Коллинза-Сассмана была начата разработка
- Впоследствие множество активных разработчиков присоединилось к улучшению Subversion
- 31 августа 2001 команда перешла на Subversion для управления версиями собственного исходного кода

ОСНОВНЫЕ ПОНЯТИЯ SUBVERSION

- *Хранилище* – центр хранение данных и файлов
- *Рабочая копия* Subversion - обычное дерево каталогов на вашем компьютере, содержащее набор файлов из хранилища
- *Служебный каталог рабочей копии (.svn)* – вспомогательный каталог Subversion, помогает определить, какие файлы копии содержат неоопубликованные изменения, и какие файлы устарели по отношению к файлам других источников

ХРАНЕНИЕ ИЗМЕНЕНИЙ

Дельта-кодирование (Delta encoding) — способ представления данных в виде разницы (дельты) между последовательными данными вместо самих данных.



КАК РАБОЧИЕ КОПИИ ОТСЛЕЖИВАЮТ ХРАНИЛИЩЕ

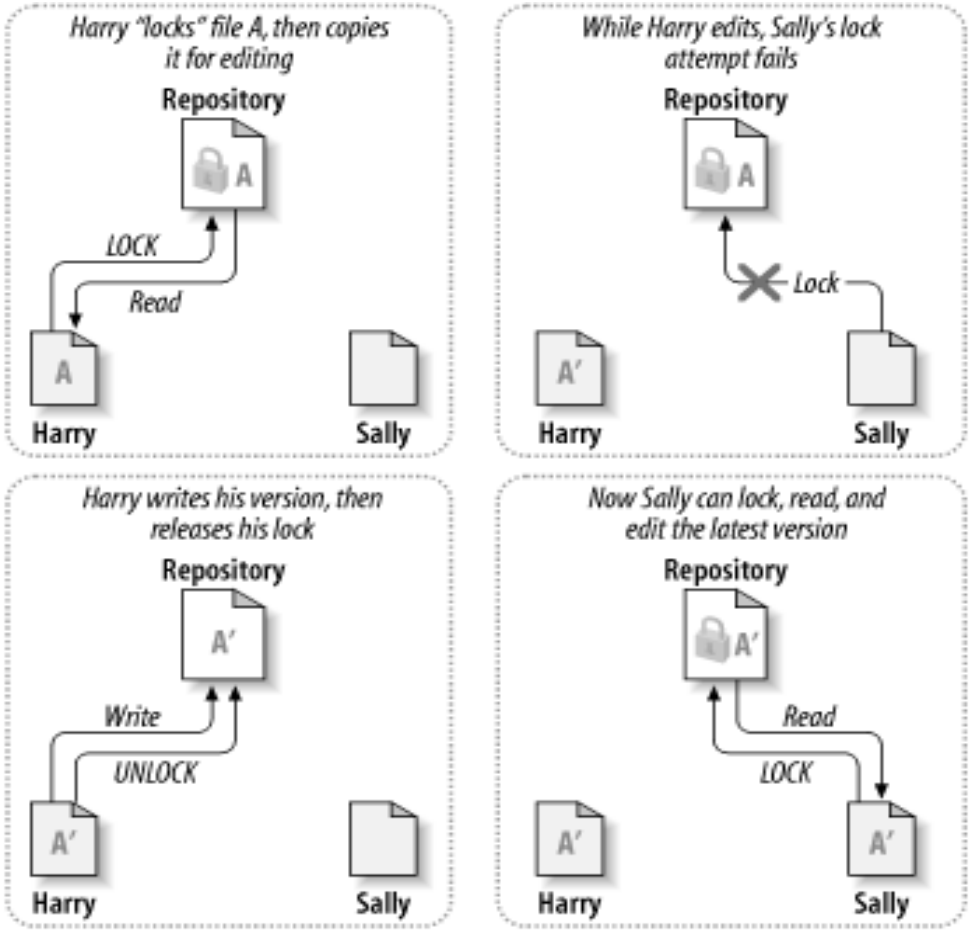
Используя информацию из локального репозитория и хранилища, можно сказать, в каком из четырёх состояний находится рабочий файл:

- Не изменялся и не устарел
- Изменялся локально и не устарел
- Не изменялся и устарел
- Изменялся локально и устарел (файл необходимо сначала обновить)

МОДЕЛИ ВЕРСИОНИРОВАНИЯ

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

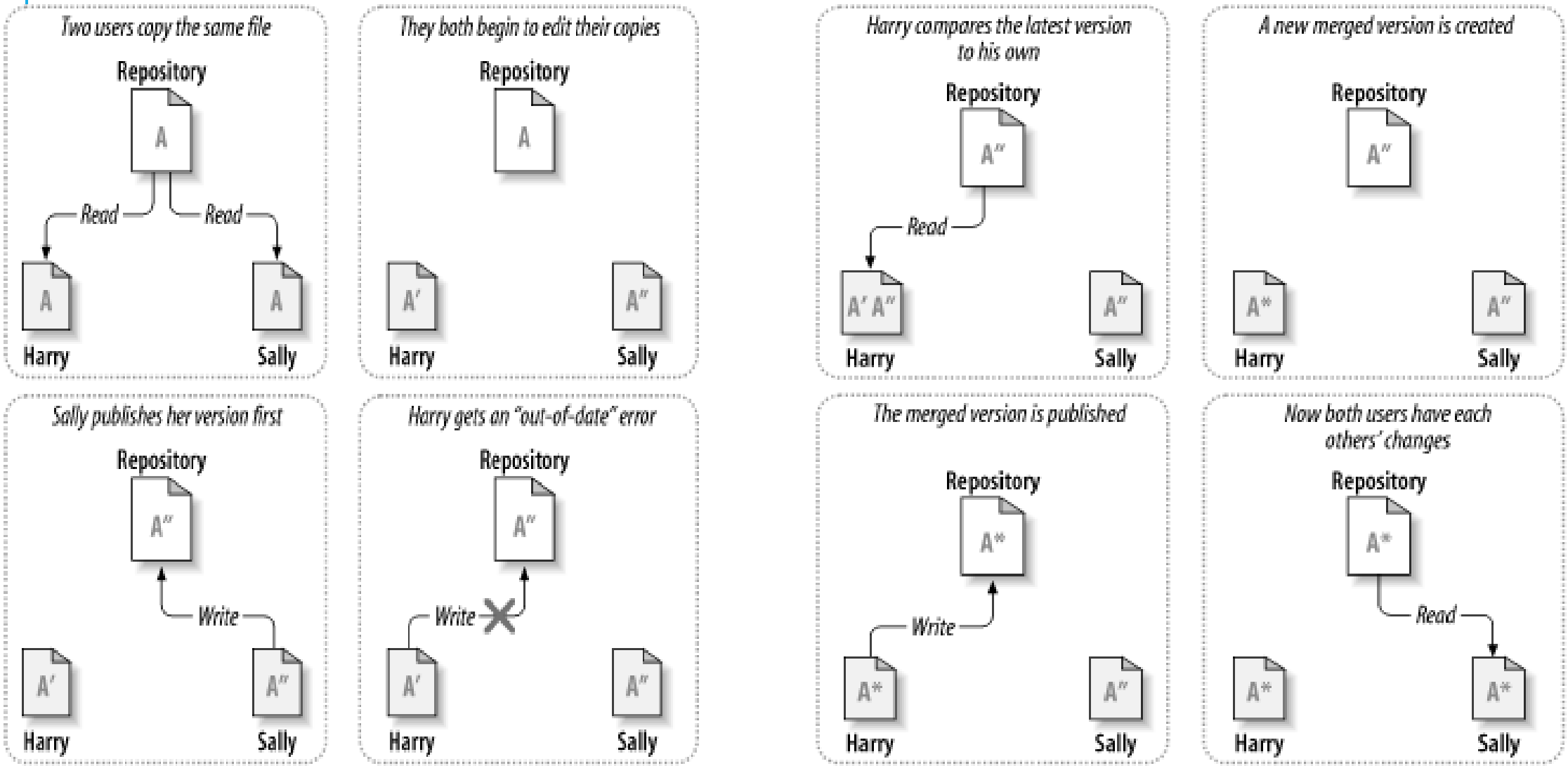
ПРИМЕР МОДЕЛИ «БЛОКИРОВАНИЕ - ИЗМЕНЕНИЕ - РАЗБЛОКИРОВАНИЕ»



ПРОБЛЕМЫ МОДЕЛИ

- Проблемы администрирования (можно забыть разблокировать файл)
- Блокирование, когда оно не нужно (при редактировании различных частей файла)
- Ложное чувство безопасности при блокировке (если файлы зависят друг от друга и вы их измените, поломки всё равно произойдут)

ПРИМЕР МОДЕЛИ «КОПИРОВАНИЕ — ИЗМЕНЕНИЕ - СЛИЯНИЕ»



ПЛЮСЫ МОДЕЛИ

- Пользователи могут работать параллельно, не тратя время на ожидание друг друга
- Редкие конфликты в файлах, так как зачастую изменения не перекрываются
- Время, которое было потрачено на разрешение конфликтов значительно меньше времени отнимаемого блокирующей системой.

ПОЛЕЗНЫЕ КОМАНДЫ

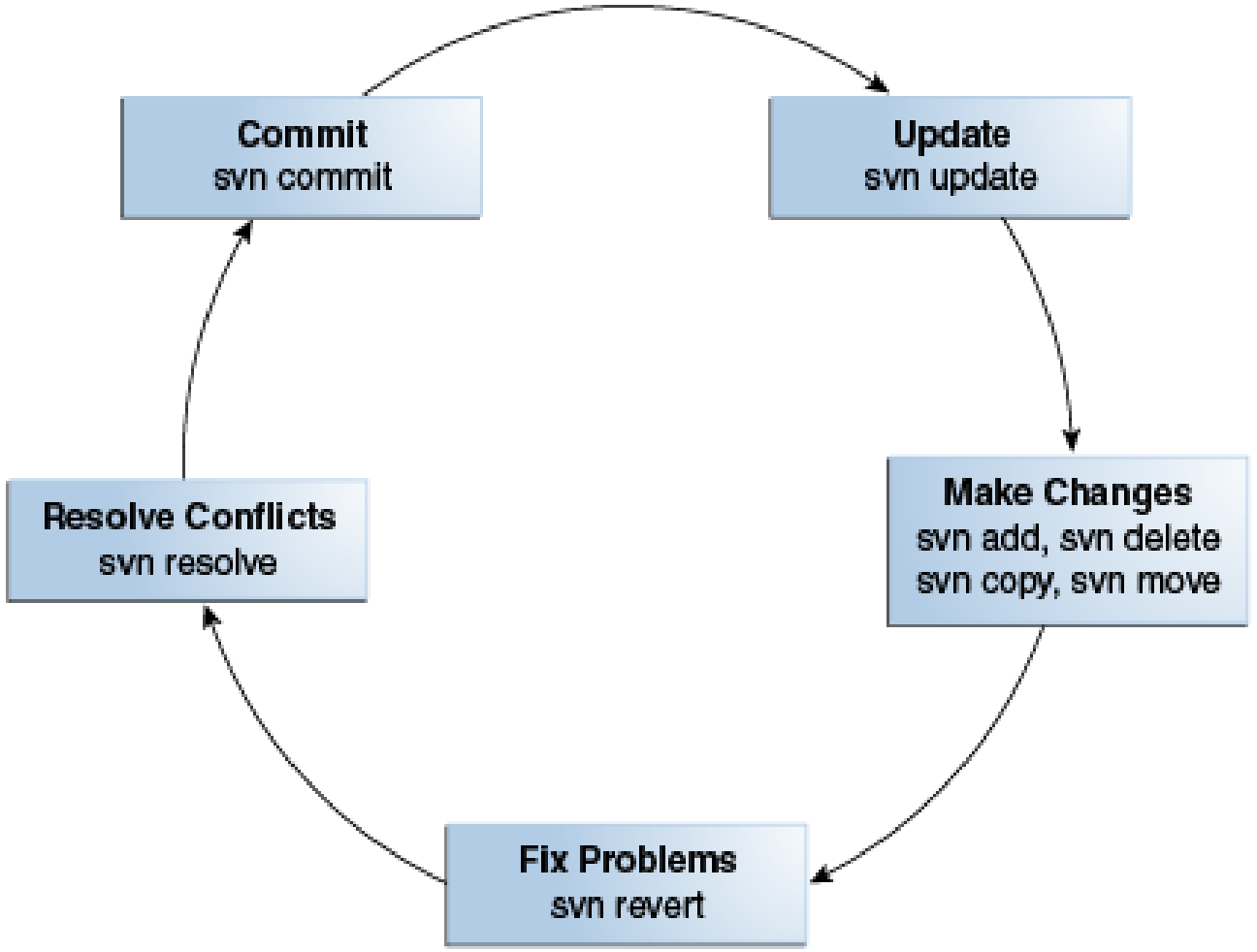
svn help

команда *svn help <subcommand>* покажет описание синтаксиса, параметров и поведения подкоманды subcommand

svn info

информация о репозитории, такая как URL родительского репозитория, номер текущей ревизии, дата последних изменений

ЖИЗНЕННЫЙ ЦИКЛ ПРОЕКТА В SUBVERSION



ФИКСАЦИЯ

Фиксаця (check-in, commit, submit) - создание новой версии, публикация изменений. Распространение изменений, сделанных в рабочей копии, на хранилище документов. При этом в хранилище создаётся новая версия изменённых документов.

1. СОЗДАНИЕ РЕПОЗИТОРИЯ ИЛИ ИМПОРТ СУЩЕСТВУЮЩЕГО

Создание пустого репозитория

$ svnadmin create /path/to/rep

Создание репозитория из исходных кодов

Для импортирования нового проекта в Subversion-хранилище используется *svn import*. При необходимости она создаёт промежуточные директории

$ svn import mytree
*file:///usr/local/svn/newrepos/some/project *

2. СОЗДАНИЕ РАБОЧЕЙ КОПИИ

Для того чтобы создать рабочую копию уже существующего репозитория, вам нужно получить какой-либо из подкаталогов хранилища. Для этого используется svn checkout.

Пример:

$ svn checkout http://svn.example.com/repos/calc

A calc/Makefile

Checked out revision 56.

URL ХРАНИЛИЩА

Получить доступ к хранилищу Subversion можно различными способами — на локальном диске или через ряд сетевых протоколов:

- file:///
- http://
- https://
- svn://
- svn+ssh://

3. ОБНОВЛЕНИЕ РЕПОЗИТОРИЯ

Используйте *svn update* для синхронизации вашей рабочей копии с последней правкой в хранилище.

$ svn update

U button.c

Updated to revision 57.

БУКВЕННЫЕ КОДЫ, ПОЛУЧАЕМЫЕ ПРИ ПОЛУЧЕНИИ ИЗМЕНЕНИЙ С СЕРВЕРА

Буквенное обозначение	Расшифровка	Описание
U	Updated	Файл был обновлён
A	Added	Файл был добавлен
D	Deleted	Файл был удалён
R	Replaced	Файл был перемещён
G	merGed	Файл был объединён
C	Conflictiong	Файл конфликтует с вашей локальной версией

4. ВНЕСЕНИЕ ИЗМЕНЕНИЙ В РАБОЧУЮ КОПИЮ

- **svn add foo** – добавление файла (каталога или символьной ссылки) foo
- **svn delete foo** – удаление foo
- **svn copy foo bar** – копирвоание foo в bar
- **svn move foo bar** – перемещение foo в bar
- **svn mkdir dir_name** - создание директории

5. ИСПРАВЛЕНИЕ ИЗМЕНЕНИЙ

Если вы обнаружили, что изменения в файле являются ошибочными и вы не хотите, чтобы они попали в репозиторий, вы можете отменить своиз изменения командой **svn revert**

Пример:

$ svn revert README

Reverted 'README'

6. ИСПРАВЛЕНИЕ КОНФЛИКТОВ

Что делает Subversion при возникновении конфликта?

- Помечает файл как конфликтный (C)
- Если файл объединяемого типа, ставит маркеры конфликта
- Добавляет в рабочую копию до трёх не версионированных файлов:

1. filename.mine
2. filename.rOLDREV (OLDREV - это номер правки файла в директории .svn)
3. filename.rNEWREV (NEWREW - номер правки HEAD хранилища)

СПОСОБЫ РАЗРЕШЕНИЯ КОНФЛИКТОВ

- Объединить конфликтующий текст «вручную» (путем анализа и редактирования маркеров конфликта в файле)
- Скопировать один из временных файлов поверх своего рабочего файла
- Выполнить *svn revert <filename>* для того, чтобы убрать все ваши локальные изменения

ВИД ФАЙЛА С МАРКЕРАМИ КОНФЛИКТА

<<<<<<< имя файла

ваши изменения

=======

версия из репозитория

>>>>>>> ревизия

7. АНАЛИЗ ИЗМЕНЕНИЙ

svn status

Находит все сделанные вами файловые и структурные изменения.

svn diff

Команда *svn diff* формирует свой вывод сравнивая ваши рабочие файлы с «нетронутыми» копиями из .svn. Весь текст запланированных для добавления файлов показывается как добавленный (+), а весь текст запланированных для удаления файлов показывается как удалённый (-)

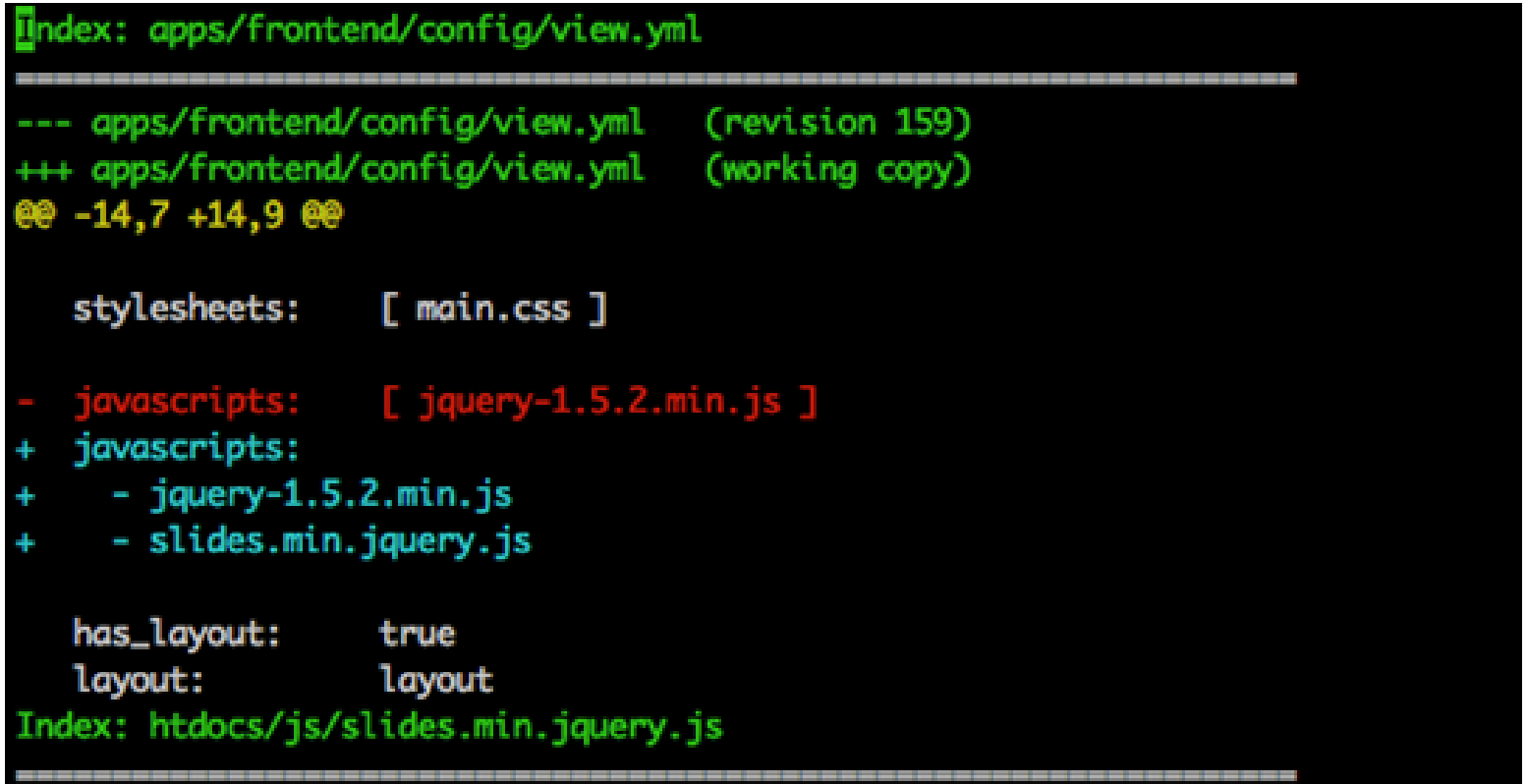
7. АНАЛИЗ ИЗМЕНЕНИЙ

Основные буквенные коды svn status:

Буквенный код	**Пояснение**
' '	Без модификаций
A	Объект запланирован для добавления
D	Объект запланирован для удаления
M	Объект был изменён
R	Объект был заменён внутри рабочей копии
C	Конфликтующий объект
X	Объект был включён внешне
I	Объект был заигнорирован
?	Объект не под версионным контролем
!	Объект не найден (был перемещён или удалён)
~	Объект поменял свой тип (файл, директория, ссылка)

7. АНАЛИЗ ИЗМЕНЕНИЙ

Увидеть, как именно вы изменили элементы, можно запустив *svn diff* без аргументов, в результате выведутся изменения файлов в виде единого формата представления различий. При этом удалённые строки предваряются -, а добавленные строки предваряются +



8. ФИКСАЦИЯ ИЗМЕНЕНИЙ

Сначала необходимо добавить файлы, которые войдут в следующую фиксацию командой svn add <path to file>

svn commit отправляет все ваши изменения в хранилище. При фиксации изменений необходимо написать лог - сообщение, оно будет присоединено к правке.

$ svn commit --message "Corrected number of cheese slices."

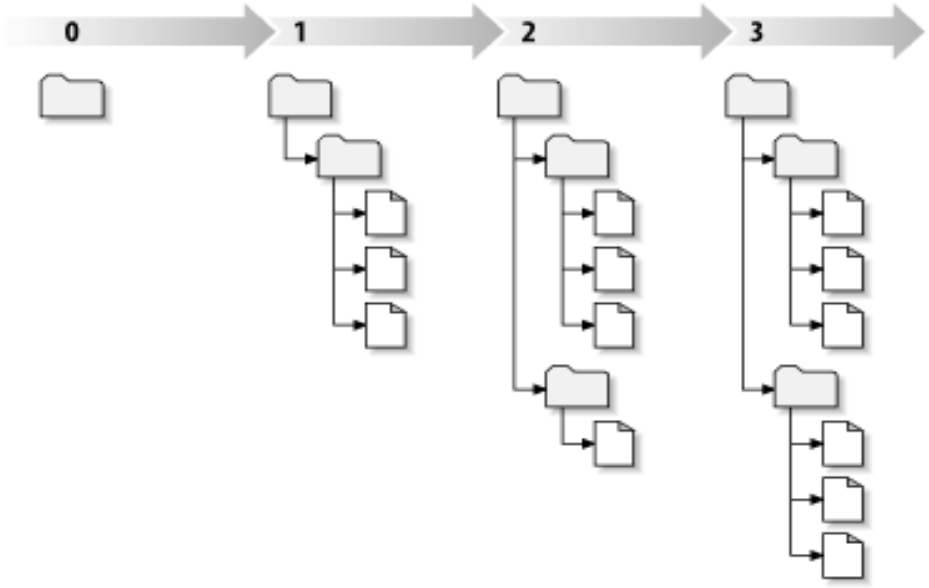
Sending sandwich.txt

Transmitting file data .

Committed revision 3.

ПРИМЕР РЕВИЗИЙ ХРАНИЛИЩА

Тогда, когда происходит фиксация, создаётся новое состояние файловой системы, которое называется *правка*. Каждая правка получает уникальный номер, на 1 больший номера предыдущей правки. Начальная правка только что созданного хранилища получает номер 0.



ПРОСМОТР ИСТОРИИ ИЗМЕНЕНИЙ

svn log

Показывает вам развернутую информацию: лог сообщения с указанной датой изменений и их автором, а также измененные пути файлов.

Пример:

$ svn log

--

r1 | sally | Mon, 15 Jul 2002 17:40:08 -0500 | 1 line

Initial import

ДРУГИЕ КОМАНДЫ ДЛЯ ПРОСМОТРА ИСТОРИИ ИЗМЕНЕНИЙ

svn cat

Если вы хотите проанализировать ранние версии файла, а не различия между двумя файлами, можно воспользоваться svn cat:

$ svn cat --revision 2 rules.txt

svn list

Команда *svn list* показывает содержимое директории в хранилище, при этом не закачивая его на локальную машину

ДРУГИЕ ПОЛЕЗНЫЕ КОМАНДЫ

svn cleanup

Перезапускает выполнение лог - файлов, помогает Subversion завершить предварительно начатые операции и рабочая копия снова вернется в согласованное состояние

КЛЮЧЕВЫЕ СЛОВА ПРАВОК

- **HEAD -** последняя (или «самая новая») правка хранилища
- **BASE -** номер правки элемента рабочей копии. Если элемент редактировался, то «BASE версия» соответствует тому, как элемент выглядел до редактирования.
- **COMMITTED -** правка, в которой элемент последний раз редактировался (предшествующая либо равная BASE)
- **PREV -** правка, предшествующая последней правке, в которой элемент был изменен

ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ КЛЮЧЕВЫХ СЛОВ ПРАВОК

$ svn diff --revision PREV:COMMITTED foo.c

 # показать последнее изменение зафиксированное для foo.c

$ svn diff --revision HEAD

 # сравнить ваш рабочый файл 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

ИГНОРИРОВАНИЕ ФАЙЛОВ

Игнорирование — процесс, позволяющий хранить файлы в репозитории не под версионным контролем. СКВ не предлагает добавить заигнорированные файлы в следующую фиксацию.

ИГНОРИРОВАНИЕ В SUBVERSION

Игнорирование — процесс, позволяющий хранить файлы в репозитории не под версионным контролем. СКВ не предлагает добавить заигнорированные файлы в следующую фиксацию.

Глобальное игнорирование

Локальное игнорирование внутри проекта

Например, локальное игнорирование файлов с расширением jpg:

svn propset svn:ignore ".jpg" .*

Для просмотра игнорированных файлов, необходима команда svn status с ключом — no-ignore.

svn status --no-ignore | grep "^I"

I myimage.jpg