Профессиональный практикум программной инженерии

Лекция №2: «Системы контроля версий»

Грищенко Виктор Игоревич

<victor.grischenko@gmail.com>

Зачем нужны системы контроля версий (СКВ)?

- Сохранение истории изменений проекта
- Поддержка нескольких веток разработки
- Упрощение коллективной работы с кодом
- Устранение «боязни внесения изменений»

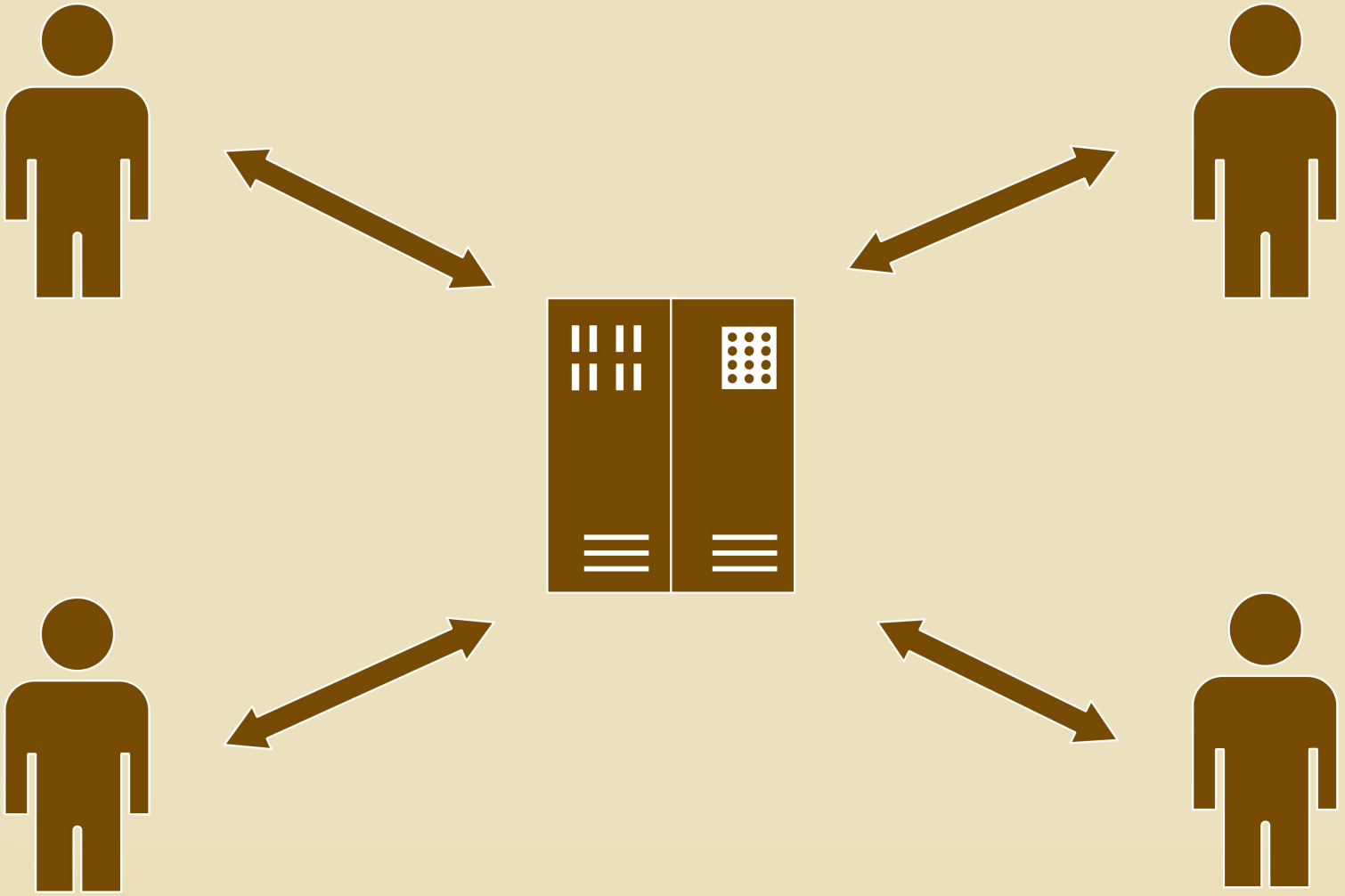
Основные типы СКВ

- Централизованные
  - CVS
  - SVN
- Распределенные
  - GIT
  - Bazaar
  - Mercurial

Централизованные системы контроля версий

- Один сервер с репозиторием
- Работа через сеть
- Простой контроль над процессом разработки
- Простая организация резервного копирования

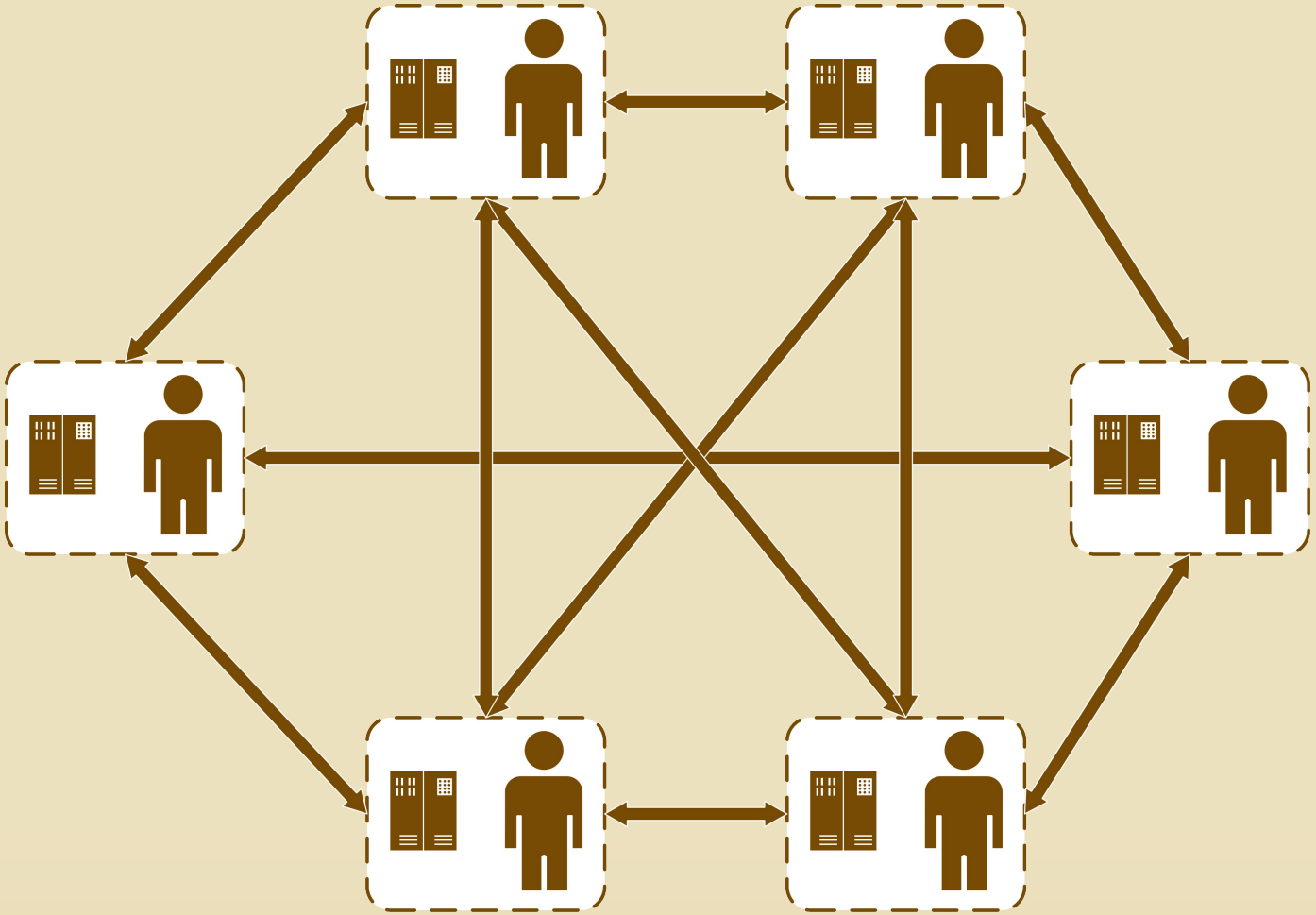
Централизованные системы контроля версий



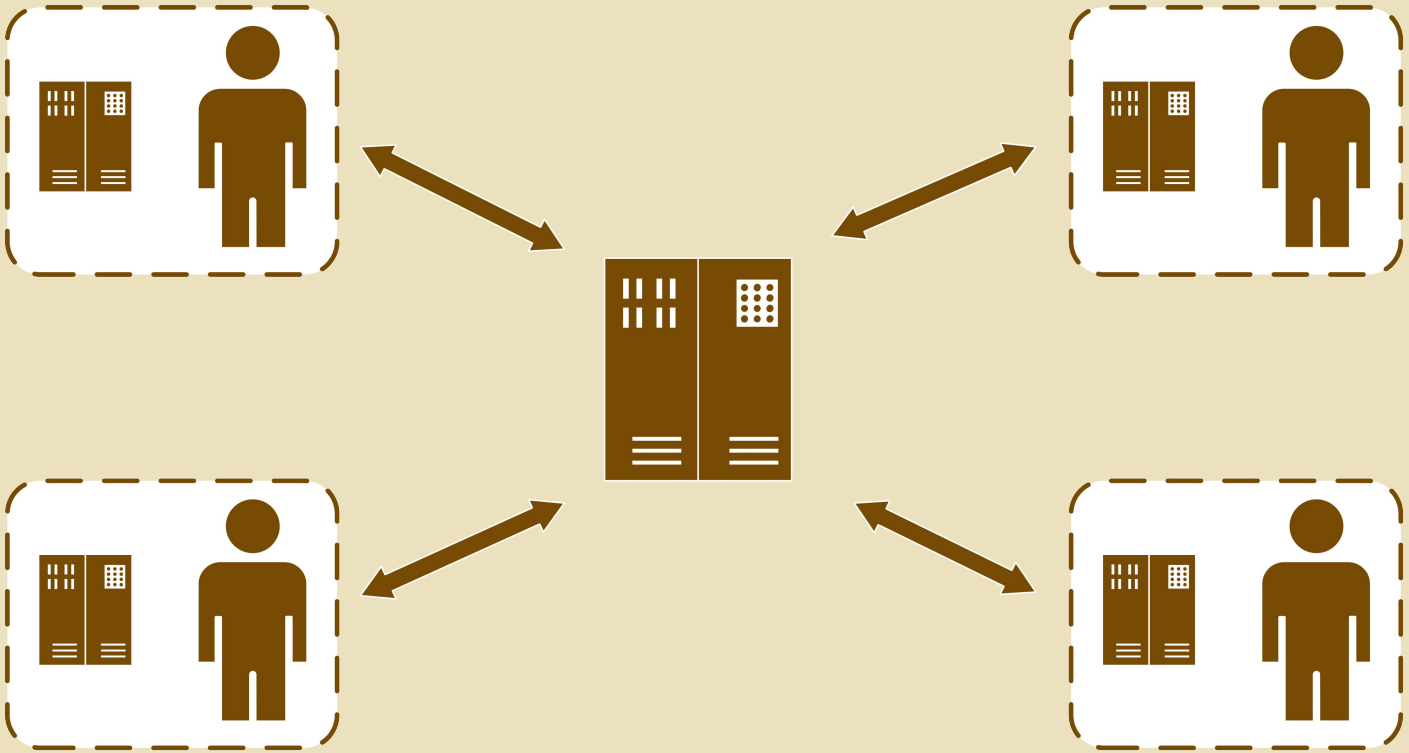
Распределенные системы контроля версий

- Каждая рабочая копия — полная версия репозитория
- Работа с локальным репозиторием
- Отсутствие единого централизованного репозитория

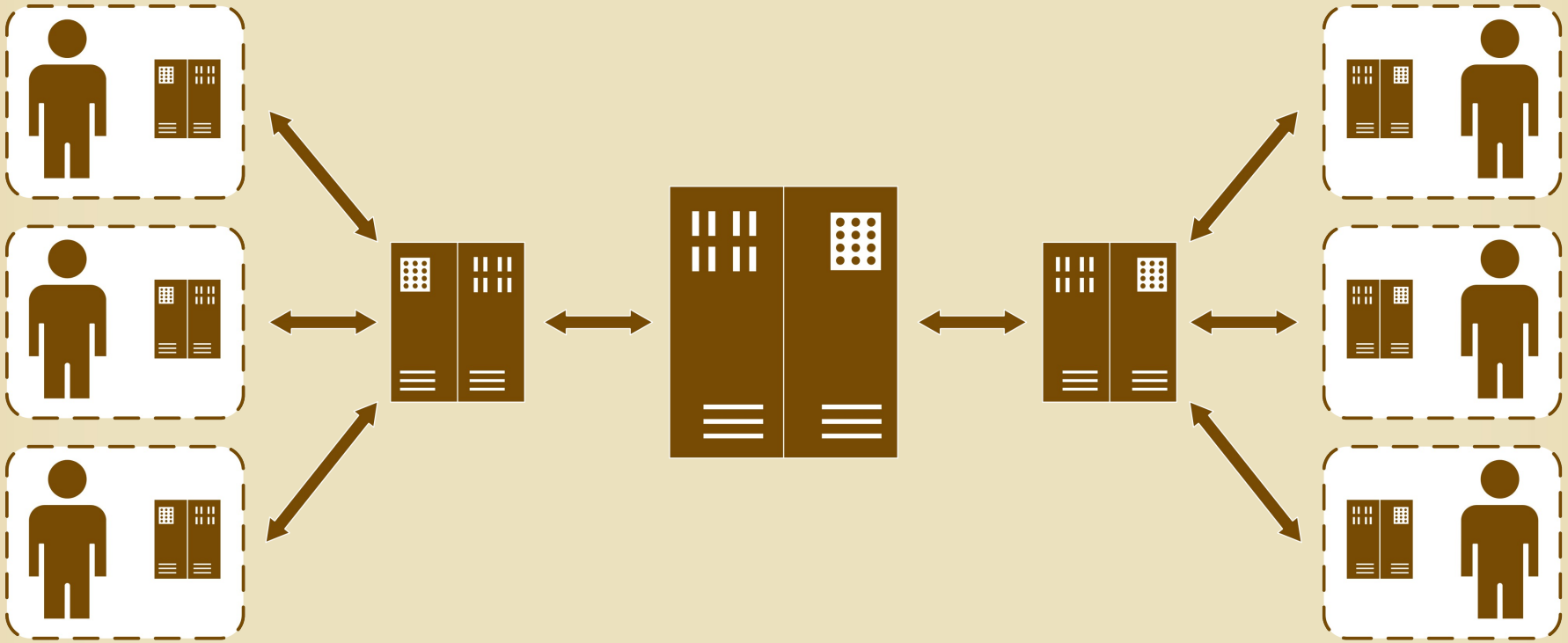
Распределенные системы контроля версий



Распределенные системы контроля версий



Распределенные системы контроля версий



Базовый цикл разработки с использованием SVN

- Создание репозитория
- Первичный внесение данных
- Создание рабочей копии
- Внесение изменений в рабочую копию
- Фиксация изменений

Структура репозитория

- trunk
- branches
- tags

Создание репозитория

svnadmin create /srv/svn/repos/<имя_репозитория>

chown -R wwwrun.www /srv/svn/repos/<имя_репозитория>

cd ~/

mkdir repo

cd repo

mkdir trunk branches tags

svn import . http://svn.example.com/<имя_репозитория> -m "Создание структуры репозитория"

Первичный внесение данных

- Создание структуры проекта

  cd <каталог_с_проектом>

  svn import .
  http://svn.example.com/<имя_репозитория>/trunk -m
  "Первичное внесение данных"

Создание рабочей копии

cd <каталог_рабочей_копии>

svn checkout
http://svn.example.com/<имя_репозитория>/trunk .

A myproject/foo.c

A myproject/bar.c

…

Checked out revision 1.

Добавление и удаление файлов

- svn add bloo.h bloo.c
- svn del fish.c

Проверка состояния рабочей копии

```
svn status
M      bar.c          # файл изменен
?      foo.o          # svn не управляет foo.o
!      some_dir       # файл удален без SVN
D      fish.c         # файл удален
A      bloo.h         # файл добавлен
```

Анализ изменений в рабочей копии

```
svn diff

Index: bar.c
===============
--- bar.c (revision 3)
+++ bar.c (working copy)
@@ -1,7 +1,12 @@
 int main(void) {
-  printf("Sixty-four slices of American Cheese...\n");
+  printf("Sixty-five slices of American Cheese...\n");
 return 0;
 }
```

Фиксация изменений

cd <каталог_рабочей_копии>

svn commit -m "Комментарий к репозиторию"

Sending foo.c

Transmitting file data .

Committed revision 5.

Обновление рабочей копии

```
svn update
U    button.c
Updated to revision 57.
```

Игнорирование файлов

- Наличие в рабочей копии временных файлов

  svn status

  ? logs/access.log

  ? logs/errors.log

  ? logs/data-failes.log

Игнорирование файлов

- Создаем файл logs/.svnignore
- Указываем в нем две строки:
  - .svnignore
  - *
- Выполняем команду
  - svn propset svn:ignore . -F .svnignore

Проверка журнала изменений

svn log

---

r20 | harry | 2003-01-17 22:56:19 -0600 (Fri, 17 Jan 2003) | 1 line

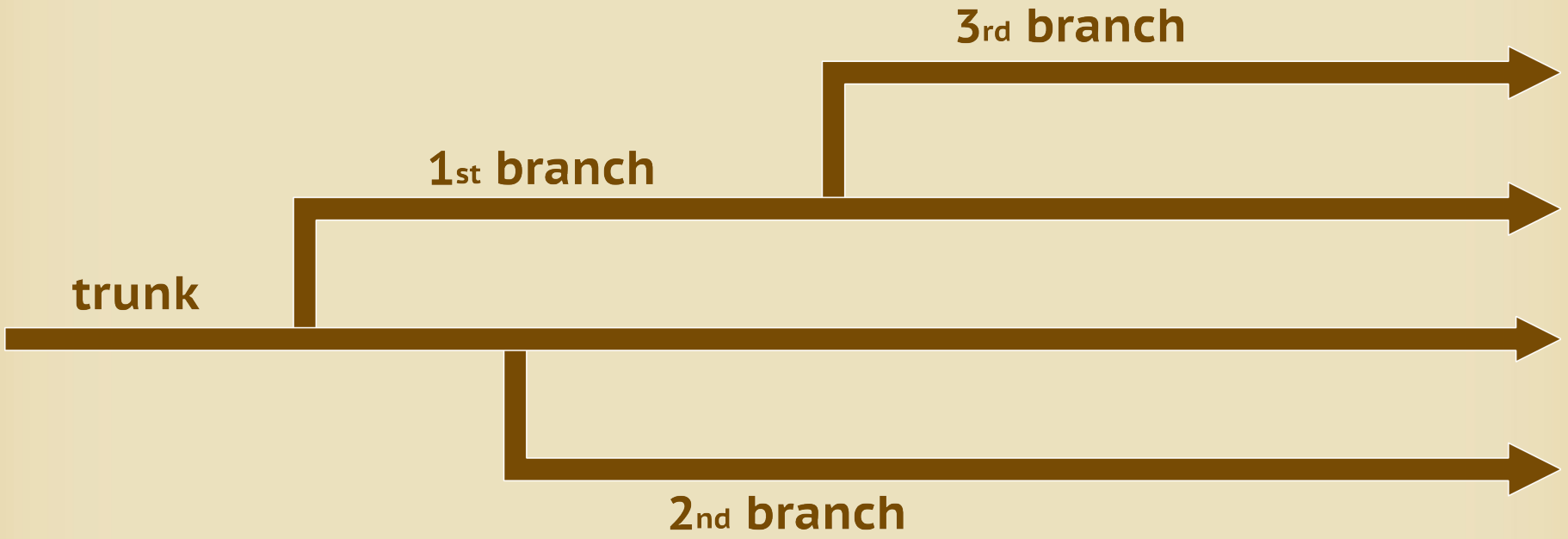
Tweak.

---

r17 | sally | 2003-01-16 23:21:19 -0600 (Thu, 16 Jan 2003) | 2 lines

...

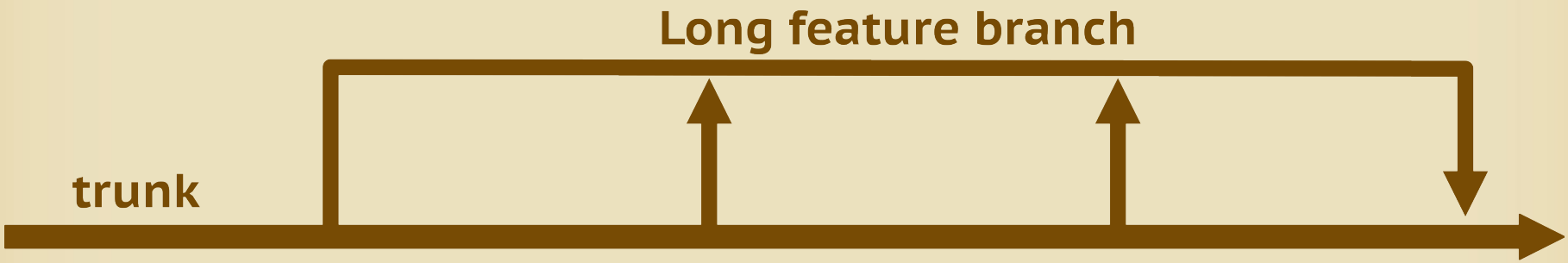
Ветки в системах контроля версий



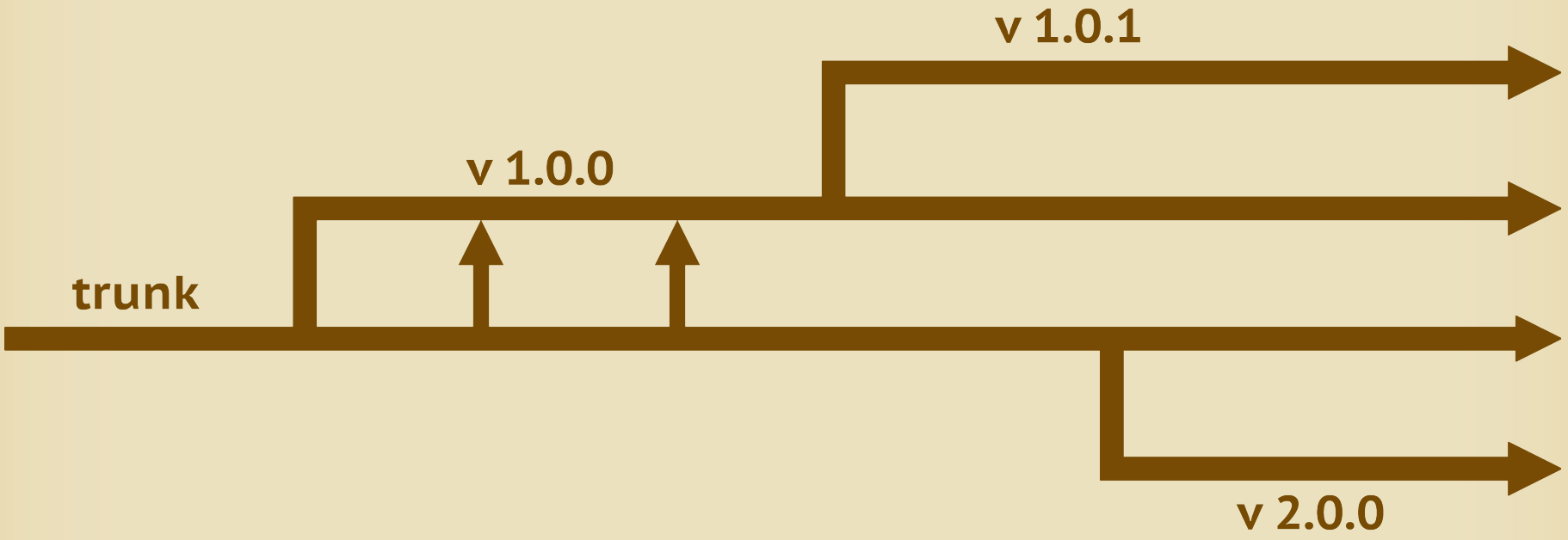
Использование ветвления

- Внесение значительных изменений в проект
- Ведение параллельной разработки нескольких версий проекта
- Рефакторинг/эксперименты

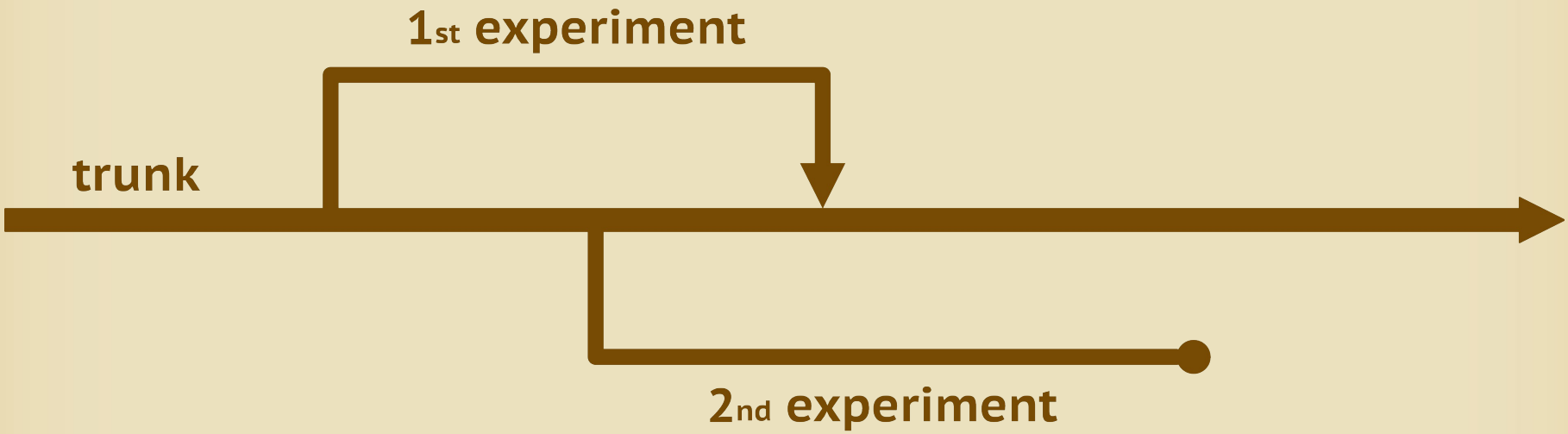
Внесение значительных изменений в проект



Параллельная разработка



Рефакторинг/эксперименты



Работа с ветками

- Создание ветки
- Переключение на ветку
- Работа в ветке
- Внесение изменений из ветки в trunk
- Удаление ветки

Создание ветки

svn copy http://svn.example.com/<имя_репозитория>/trunk http://svn.example.com/<имя_репозитория>/branches/new-branch -m "создание ветки new-branch"

Committed revision 341.

Переключение на ветку

svn switch http://svn.example.com/<имя_репозитория>/branches/new-branch

U integer.c

U button.c

U Makefile

Updated to revision 341.

Синхронизация ветки с изменениями из trunk

```
svn merge http://svn.example.com/<имя_репозитория>/trunk
U  tiny.txt
U  thhgttg.txt
U  win.txt
U  flo.txt
```

Внесение изменений из заданных ревизий

svn merge -r 4:7 http://svn.example.com/<имя_репозитория>/trunk

U tiny.txt

U thhgttg.txt

Объединение ветки с trunk

svn switch http://svn.example.com/<имя_проекта>/trunk

svn merge http://svn.example.com/<имя_проекта>/branches/new-branch

svn commit -c "Объединение с веткой new-branch"

svn delete http://svn.example.com/<имя_проекта>/branches/new-branch -c "Удаление ветки new-branch"

Конфликты при объединении и обновлении

- При объединении или обновлении рабочей копии возможно возникновение конфликтов

  svn update

  C bar.c

  Updated to revision 46.

Конфликты при объединении и обновлении

Tomato
Provolone
<<<<<<< .mine
Salami
Mortadella
Prosciutto
=======
Sauerkraut
Grilled Chicken
>>>>>>> .r2
Creole Mustard

Разрешение конфликтов

- Разрешаем вручную все конфликты во всех файлах
- Отмечаем в SVN все конфликты как разрешенные
- svn resolved bar.c

Отмена зафиксированных изменений

svn merge -r 303:302 http://svn.example.com/<имя_репозитория>/trunk

svn commit -m "Отмена ошибочных изменений"

Типовые примеры использования веток

- Функциональные ветки (branches)
- Метки (tags)
- Поддержка выпуска релизов

Поддержка выпуска релизов

- trunk – основная ветка разработки
- branches/1.0 – ветка разработки первого релиза
- branches/2.0 – ветка разработки второго релиза
- tags/1.0.0 – фиксация релиза 1.0.0
- tags/1.0.1 – фиксация корректирующего релиза 1.0.1

Что дают системы контроля версий?

- Коллективная разработка
- Автоматическое резервное копирование
- Ведение нескольких версий проекта
- Безопасный рефакторинг
- Определение ответственности за внесенные изменения
- Простое возвращение на любую версию проекта в прошлом