

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»

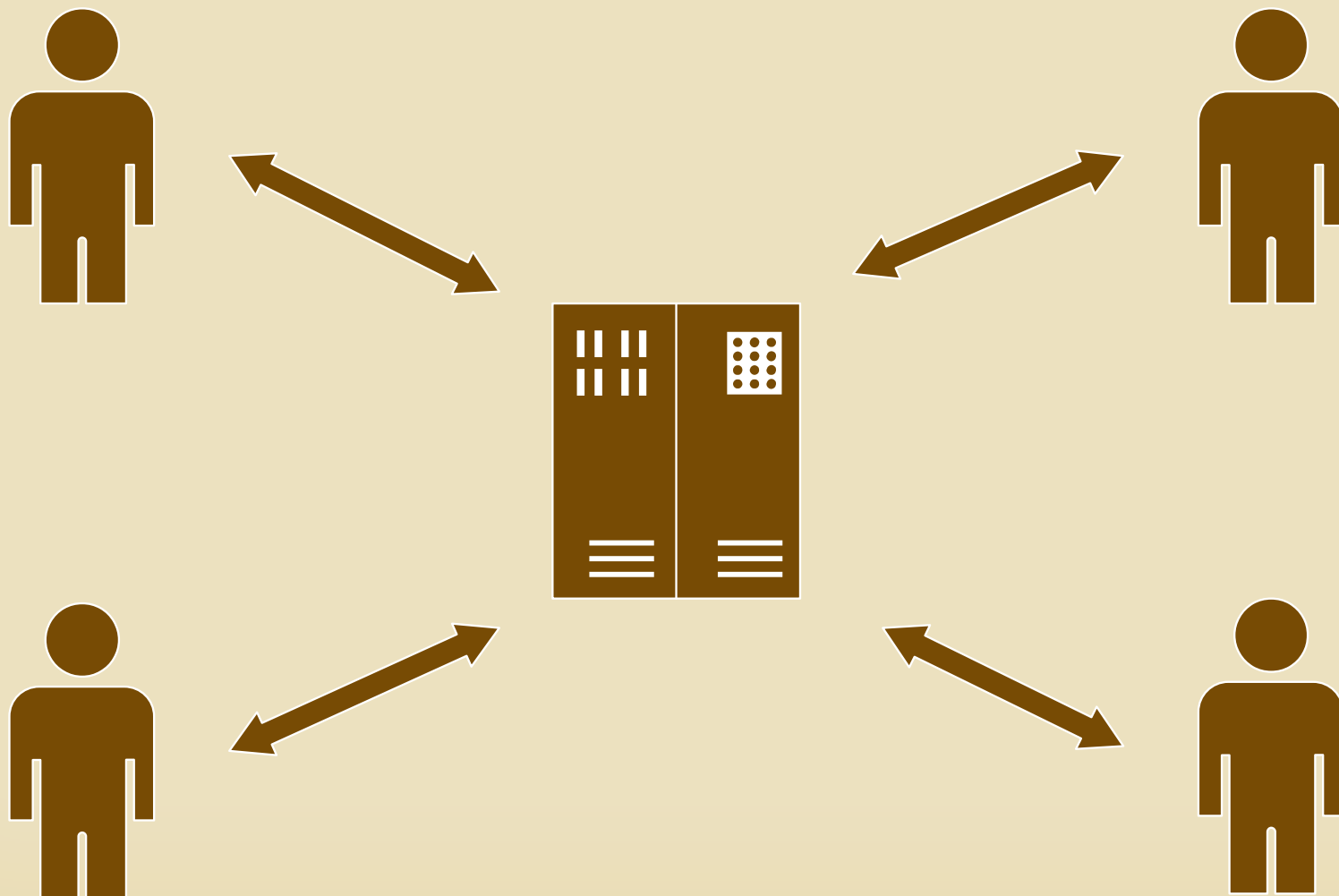
Основные типы СКВ

- 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

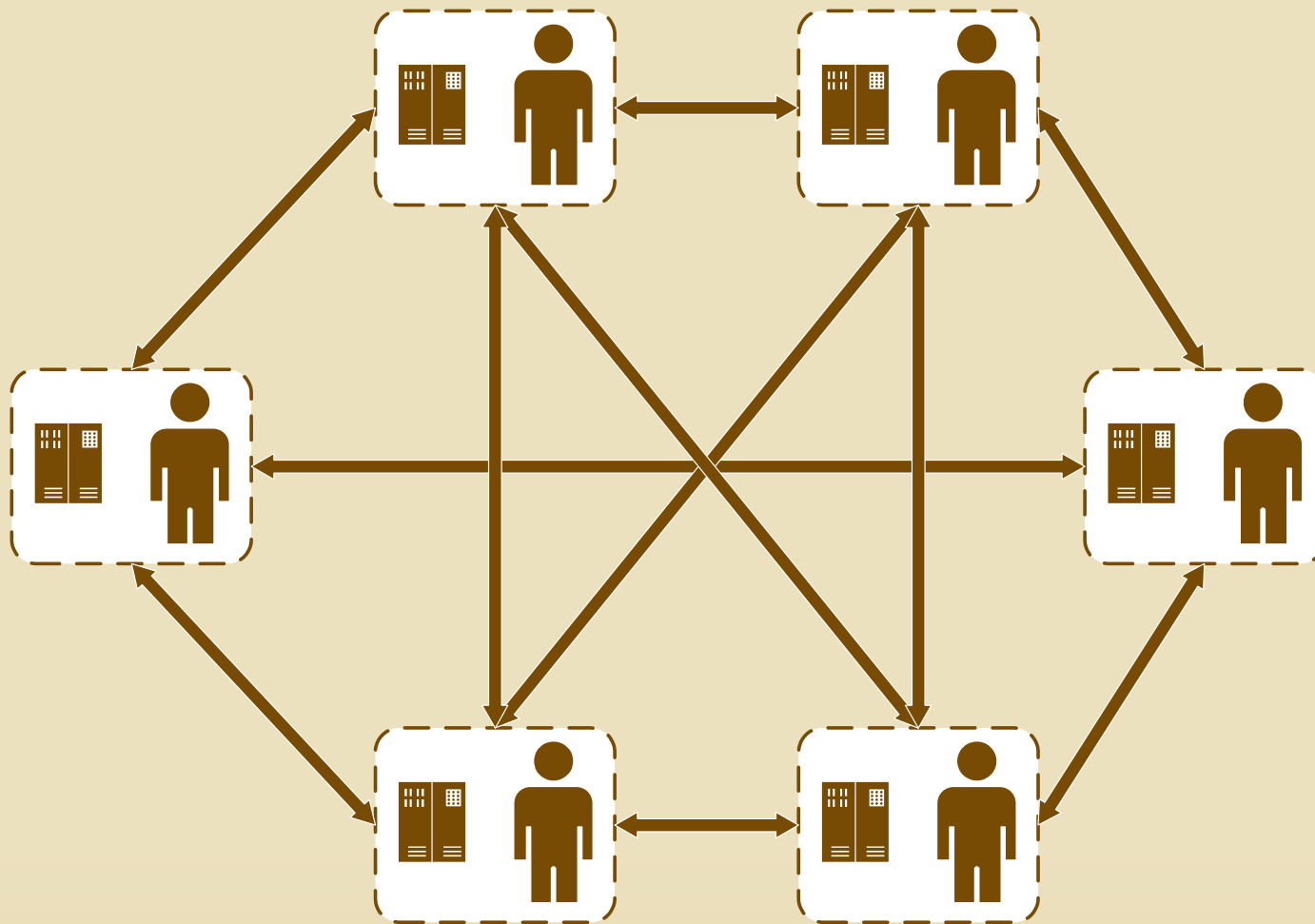
Централизованные системы контроля версий



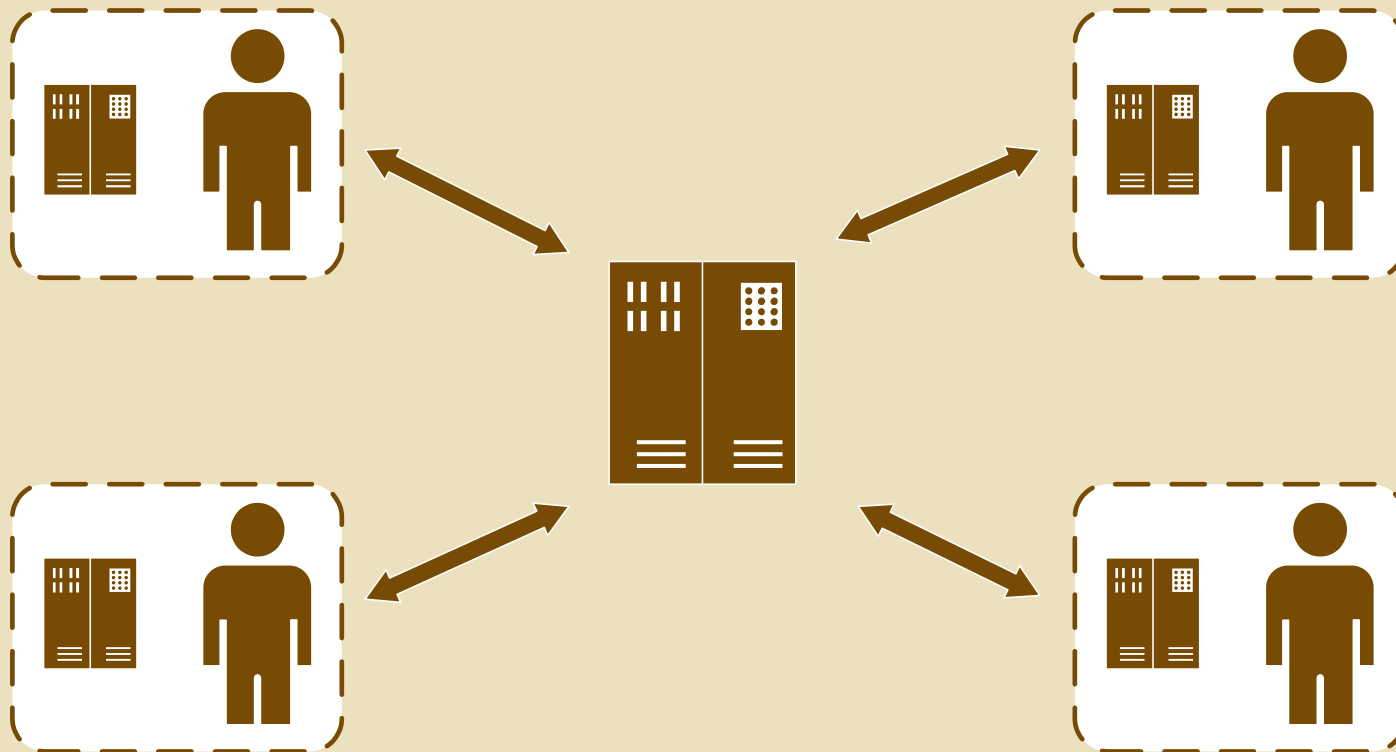
Распределенные системы контроля версий

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

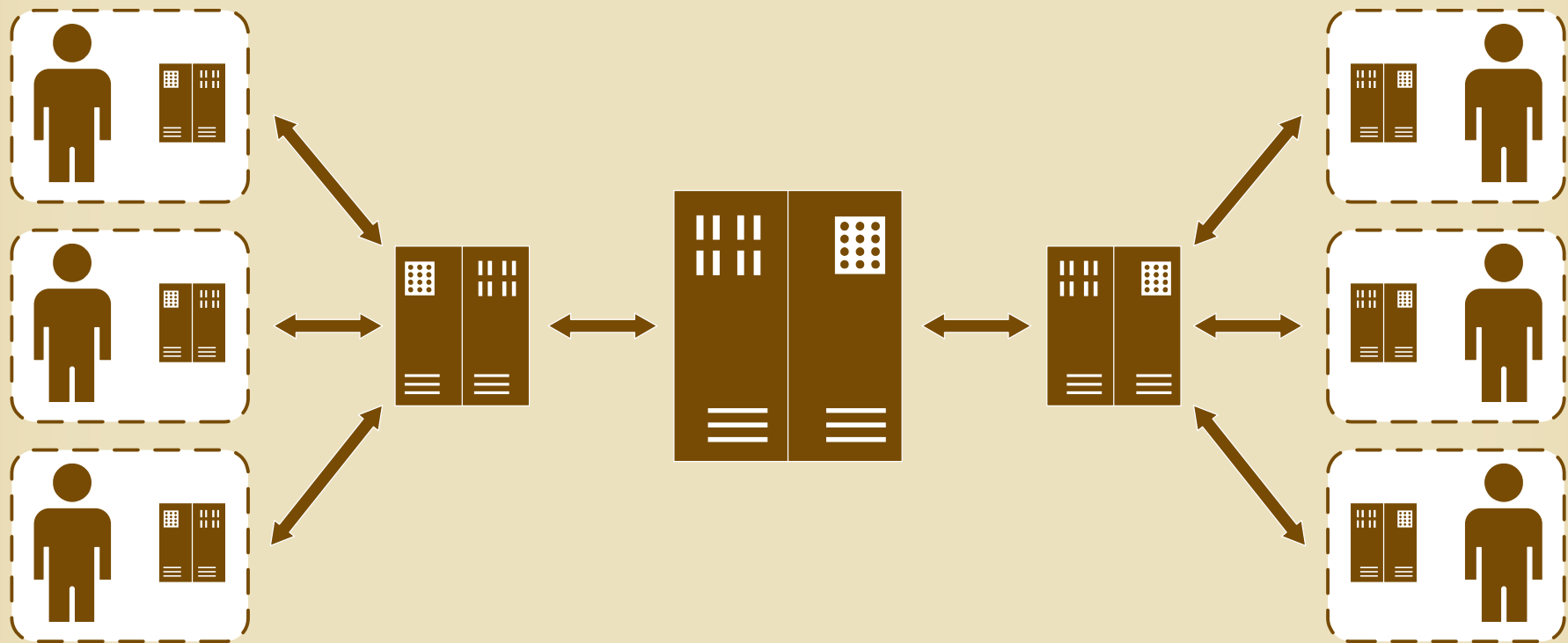
Распределенные системы контроля версий



Распределенные системы контроля версий



Распределенные системы контроля версий



Базовый цикл разработки с использованием GIT

- Настройка глобальных параметров
- Инициализация/клонирование репозитория
- Первичный внесение данных
- Внесение изменений в рабочую копию
- Фиксация изменений
- Синхронизация с удаленным репозиторием*

Настройка глобальных параметров

- > `git config --global user.name "Victor Grischenko"`
- > `git config --global user.email "victor.grischenko@gmail.com"`

Инициализация/клонирование репозитория

- Инициализация репозитория

```
> cd <каталог_с_проектом>  
> git init  
> git add .  
> git commit -m "Initial data import"
```

- Клонирование репозитория

```
> mkdir <каталог_для_проекта>  
> cd <каталог_для_проекта>  
> git clone https://github.com/IPO-14b/grischenko.git .
```

Добавление и удаление файлов

- > git add add bloo.h bloo.c
- > git rm fish.c

Проверка состояния рабочей копии

```
> git status
```

изменено	bar.c	# файл изменен
удалено	fish.c	# файл удален
новый файл	bloo.h	# файл добавлен

неотслеживаемые файлы:

foo.o	# git не управляет foo.o
-------	--------------------------

Анализ изменений в рабочей копии

```
> git diff
diff --git a/bar.c b/bar.c
index e7a5cce..77edb00 100644
=====
--- bar.c
+++ bar.c
@@ -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>
```

```
> git commit -m "Комментарий к коммиту"
```

```
[master caf5b20] Комментарий к коммиту
```

```
1 file changed, 1 insertion(+), 1 deletion(-)
```

```
create mode 100644 123
```


Выгрузка в удаленный репозиторий

> git push

Подсчет объектов: 4, готово.

Delta compression using up to 8 threads.

Сжатие объектов: 100% (4/4), готово.

Запись объектов: 100% (4/4), 5.29 MiB | 2.26 MiB/s, готово.

Total 4 (delta 0), reused 0 (delta 0)

To <https://github.com/IPO-14a/grischenko.git>

fbc2c14..20e14d5 master -> master

Загрузка из удаленного репозитория

```
> git pull
```

```
remote: Counting objects: 3, done.
```

```
remote: Compressing objects: 100% (2/2), done.
```

```
remote: Total 3 (delta 1), reused 3 (delta 1), pack-reused 0
```

```
Распаковка объектов: 100% (3/3), готово.
```

```
Из https://github.com/IPO-14a/grischenko
```

```
20e14d5..10fd3a2 master -> origin/master
```

```
Обновление 20e14d5..10fd3a2
```

```
Fast-forward
```

```
README.md | 3 ++-
```

```
1 file changed, 2 insertions(+), 1 deletion(-)
```

Проверка журнала изменений

```
> git log
```

```
commit 10fd3a2ab9f53ea5b6180653a225d0e90db61d08
```

```
Author: Victor Grischenko <victor.grischenko@gmail.com>
```

```
Date: Thu Mar 23 23:10:44 2017 +0300
```

```
test2
```

```
commit 20e14d5fffd6f9e761d2adb1f4e647edfa7a50b2
```

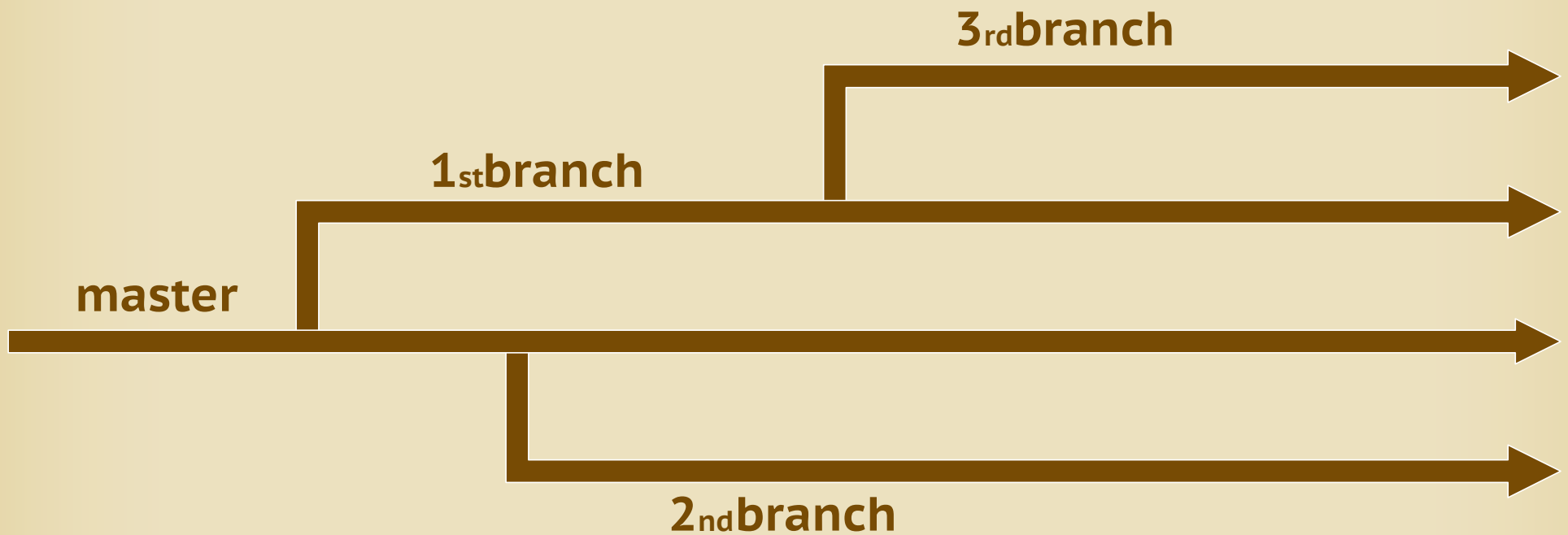
```
Author: Victor Grischenko <victor.grischenko@gmail.com>
```

```
Date: Thu Mar 23 23:06:14 2017 +0300
```

```
labs and method added
```

-
- ...

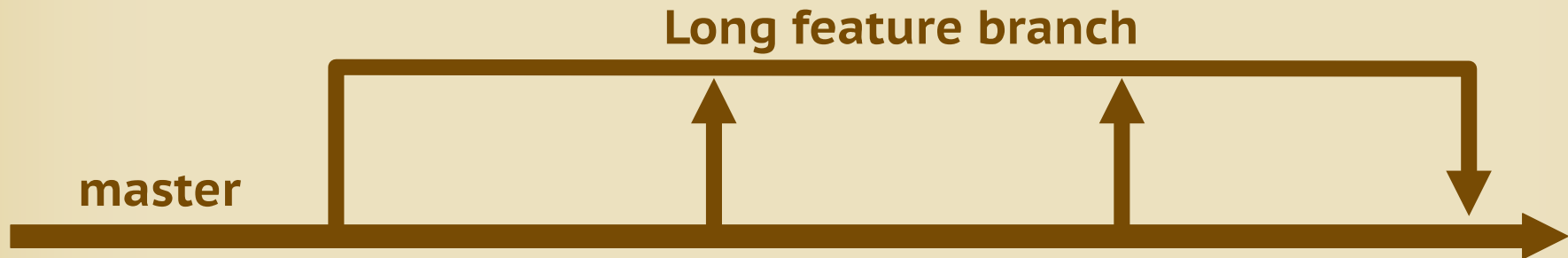
Ветки в системах контроля версий



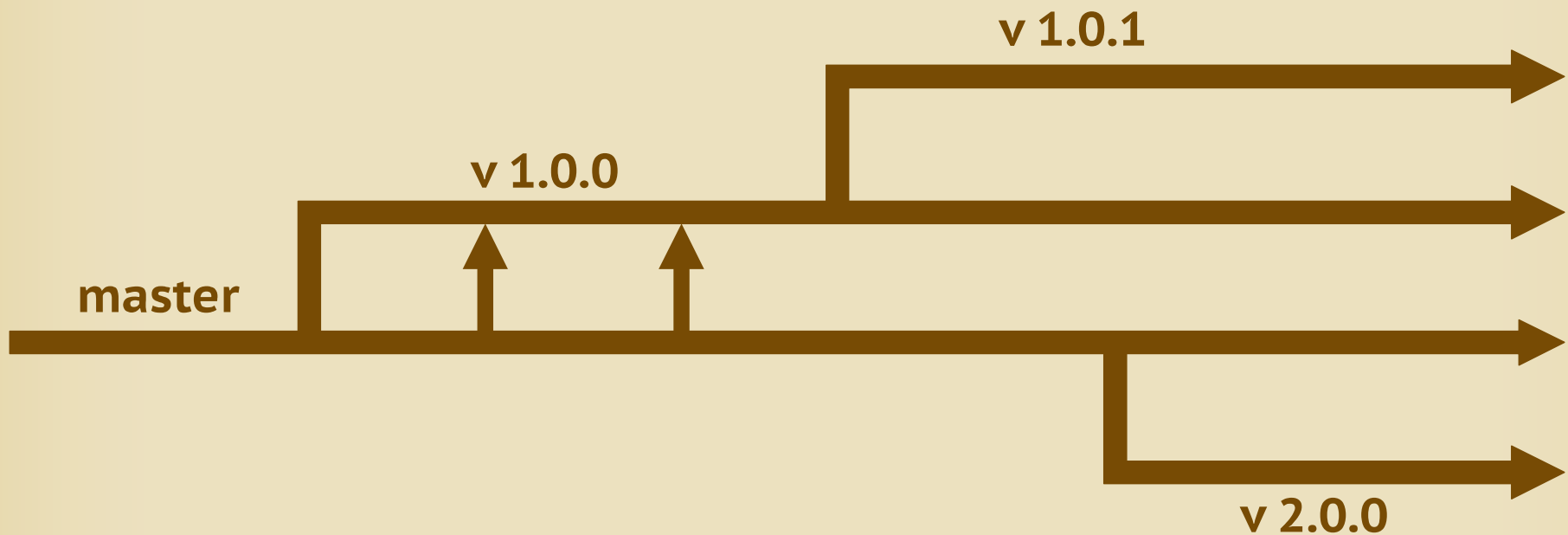
Использование ветвления

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

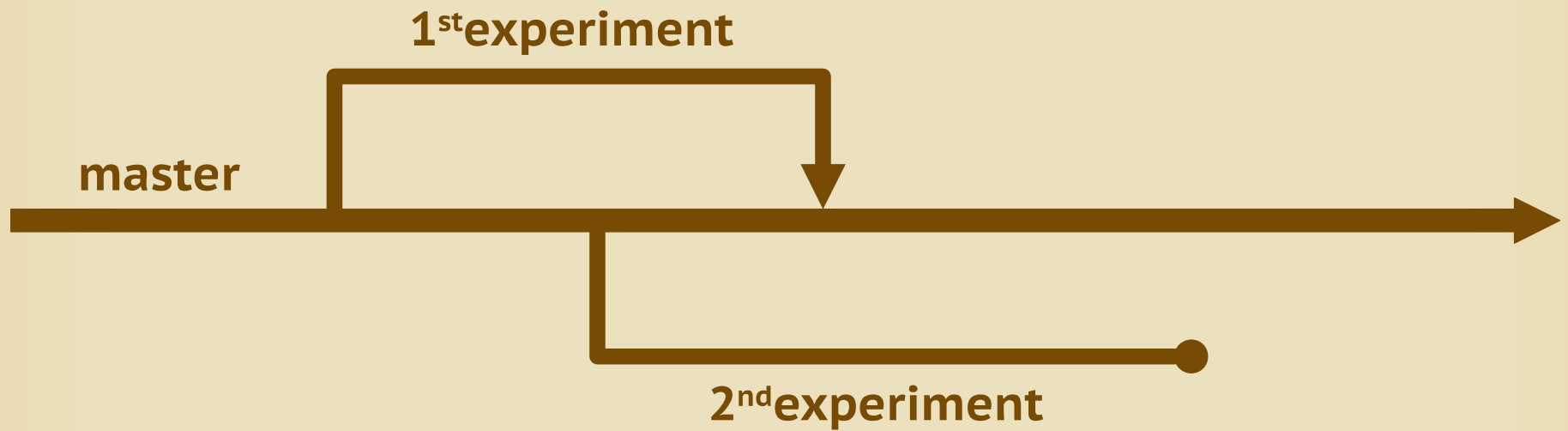
Внесение значительных изменений в проект



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

> `git checkout -b new-branch`

Переключено на новую ветку «new»

Переключение на ветку

> git co master

Переключено на ветку «master»

Ваша ветка обновлена в соответствии с «origin/master».

Синхронизация ветки с изменениями из master

> git merge master

Обновление 10fd3a2..f64f275

Fast-forward

grischenko.ppse.zip | Bin 5596300 -> 0 bytes

1 file changed, 0 insertions(+), 0 deletions(-)

delete mode 100644 grischenko.ppse.zip

Внесение изменений из заданных ревизий

```
> git cherry-pick f64f275
```

```
[master 76f74eb] test delete
```

```
Date: Thu Mar 23 23:18:43 2017 +0300
```

```
1 file changed, 0 insertions(+), 0 deletions(-)
```

```
delete mode 100644 grischenko.ppse.zip
```

Объединение ветки с master

> git checkout master

Переключено на ветку «master»

> git merge new

Обновление 10fd3a2..f64f275

Fast-forward

grischenko.ppse.zip | Bin 5596300 -> 0 bytes

1 file changed, 0 insertions(+), 0 deletions(-)

delete mode 100644 grischenko.ppse.zip

> git branch delete new

Конфликты при объединении и обновлении

> git merge test

Автослияние test.md

КОНФЛИКТ (содержимое): Конфликт слияния в test.md

Не удалось провести автоматическое слияние; исправьте конфликты и сделайте коммит результата.

Содержание конфликтного файла

Tomato

Provolone

<<<<<< HEAD

Salami

Mortadella

Prosciutto

=====

Sauerkraut

Grilled Chicken

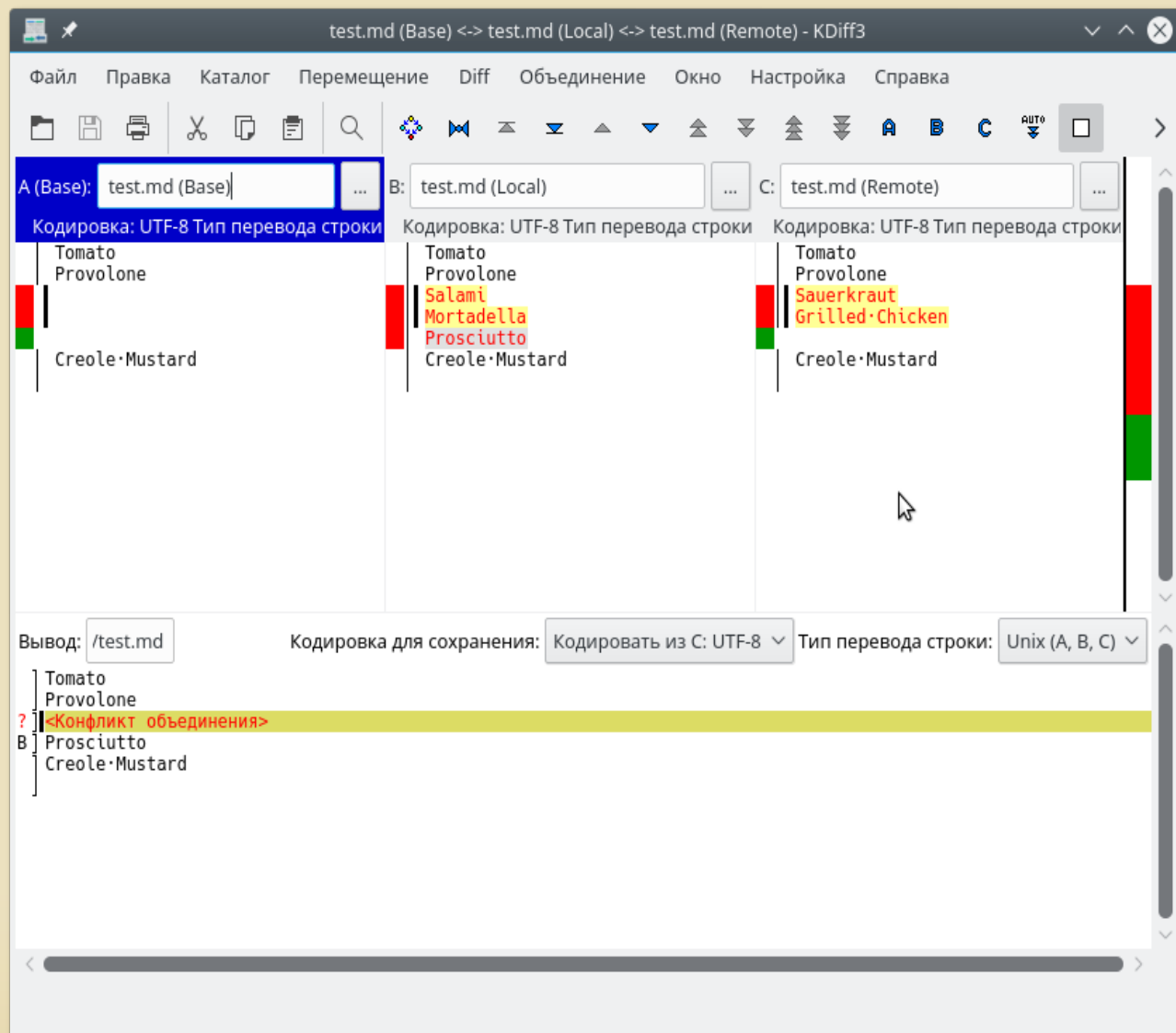
>>>>>> test

Creole Mustard

Разрешение конфликтов

- Разрешаем все конфликты во всех файлах с помощью `git mergetool`
- Зафиксировать изменения

Kdiff3



Типовые примеры использования веток

- Функциональные ветки (branches)
- Поддержка выпуска релизов

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