

Системы программирования.  
Системы контроля версий.  
Жизненный цикл ПО

Олег Французов  
2021

Что такое  
*система программирования?*

# Структура вычислительной системы

прикладные программы

**система программирования**

управление логическими устройствами

управление физическими устройствами

аппаратура

Что такое  
*программный продукт?*

*Программный продукт* – программа, которую можно использовать отчужденно от ее автора.

*Система программирования* – набор инструментальных средств для поддержки процесса разработки программного продукта в течение всего его жизненного цикла.

# Жизненный цикл программного продукта

# Жизненный цикл ПП

- Разработка
- Внедрение
- Сопровождение

- Разработка
  - Анализ требований
  - Проектирование
  - Кодирование
  - Сборка / интеграция
  - Тестирование / отладка
  - Документирование
- Внедрение
- Сопровождение

- Разработка
- Внедрение
  - Коробочный продукт
  - ПО как сервис (SaaS)
  - Разработка на заказ
  - ...
- Сопровождение

- Разработка
- Внедрение
- Сопровождение
  - Найдены дефекты
  - Изменились требования

# Анализ требований

- Бизнес-требования  
Что должен позволять делать ПП?
  - *исходят от заказчика*
- Технические требования  
Что нужно, чтобы этого добиться?
  - *формулируются разработчиками*

# Анализ требований

как он бывает на самом деле

*(видео)*

# Проектирование

- Техническое решение, удовлетворяющее требованиям
- На разных уровнях:
  - система
  - подсистема
  - модуль

# Кодирование

- Малая часть всего процесса!
- Реализация технического решения
- Конструирование *исходного кода*

*...if we wish to count lines of code, we should not regard them as 'lines produced' but as 'lines spent'*  
– Edsger Dijkstra

# Сборка / интеграция

- Превращение исходного кода в артефакты, с которыми может работать конечный пользователь
- Примеры шагов сборки:
  - компиляция и компоновка
  - построение дистрибутива
  - минификация

# Тестирование / отладка

- *Верификация*  
Продукт соответствует требованиям
- *Валидация*  
Продукт решает свои задачи

# Суть тестирования

- Компонент находится в некотором состоянии
- При определенном воздействии он должен демонстрировать ожидаемое поведение или перейти в другое ожидаемое состояние
- Задача теста: ввести компонент в исходное состояние, произвести действие, проконтролировать поведение или переход в новое состояние

# Виды тестирования

- *Модульное*  
Компонент ведет себя, как задумано
- *Интеграционное*  
Стыки между компонентами/системами
- *Пользовательское*  
Тестирование средствами  
пользовательского интерфейса
- *Нагрузочное*  
Как система ведет себя под нагрузкой?

# Виды тестирования

- Тестирование черного ящика
- Тестирование белого ящика

# Документирование

- Пользовательская документация
- Документация для разработчиков

# Модели жизненного цикла разработки

- Каскадная (водопад)
- Итерационная (agile)

история вопроса  
(*видео*)

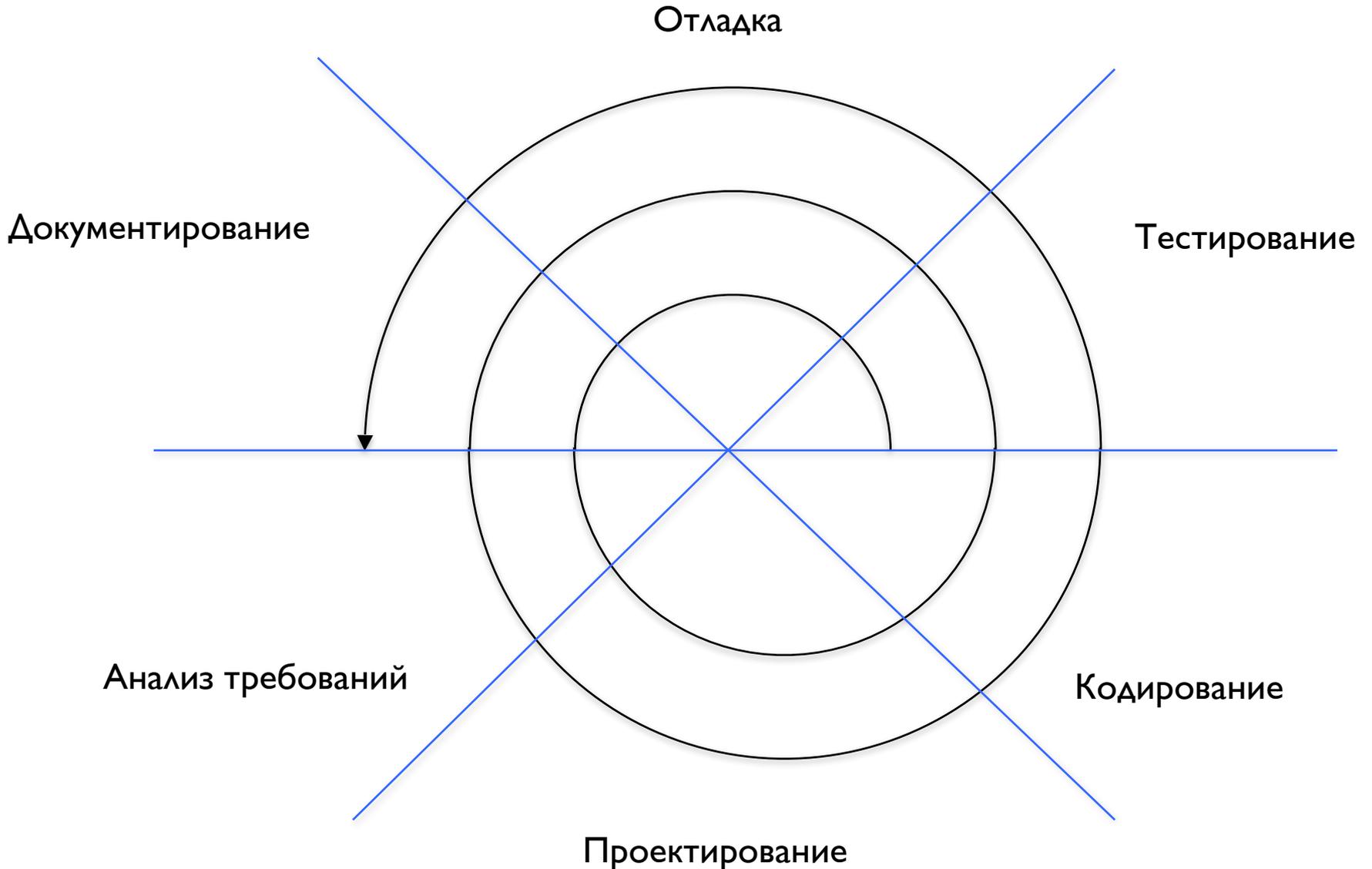
# Каскадная модель



# Каскадно-возвратная модель



# Итерационная модель



# Системы программирования

# По стратегии трансляции

- Компиляция
- Интерпретация
- Смешанная стратегия

# Примеры

- Чистая компиляция: C
- Чистая интерпретация: TCL
- Трансляция в байт-код...
  - с интерпретацией: Python
  - с JIT-компиляцией: .NET

# На примере веб-приложения

- Бэкенд: компиляция + JIT (C#)
- Шаблонизатор: интерпретация (Razor)
- CSS: трансляция (SASS)
- Фронтенд: трансляция (TypeScript)

# По степени интеграции

- Интегрированные  
Ниже порог входа
- «Кусочные»  
Выше гибкость

# Примеры

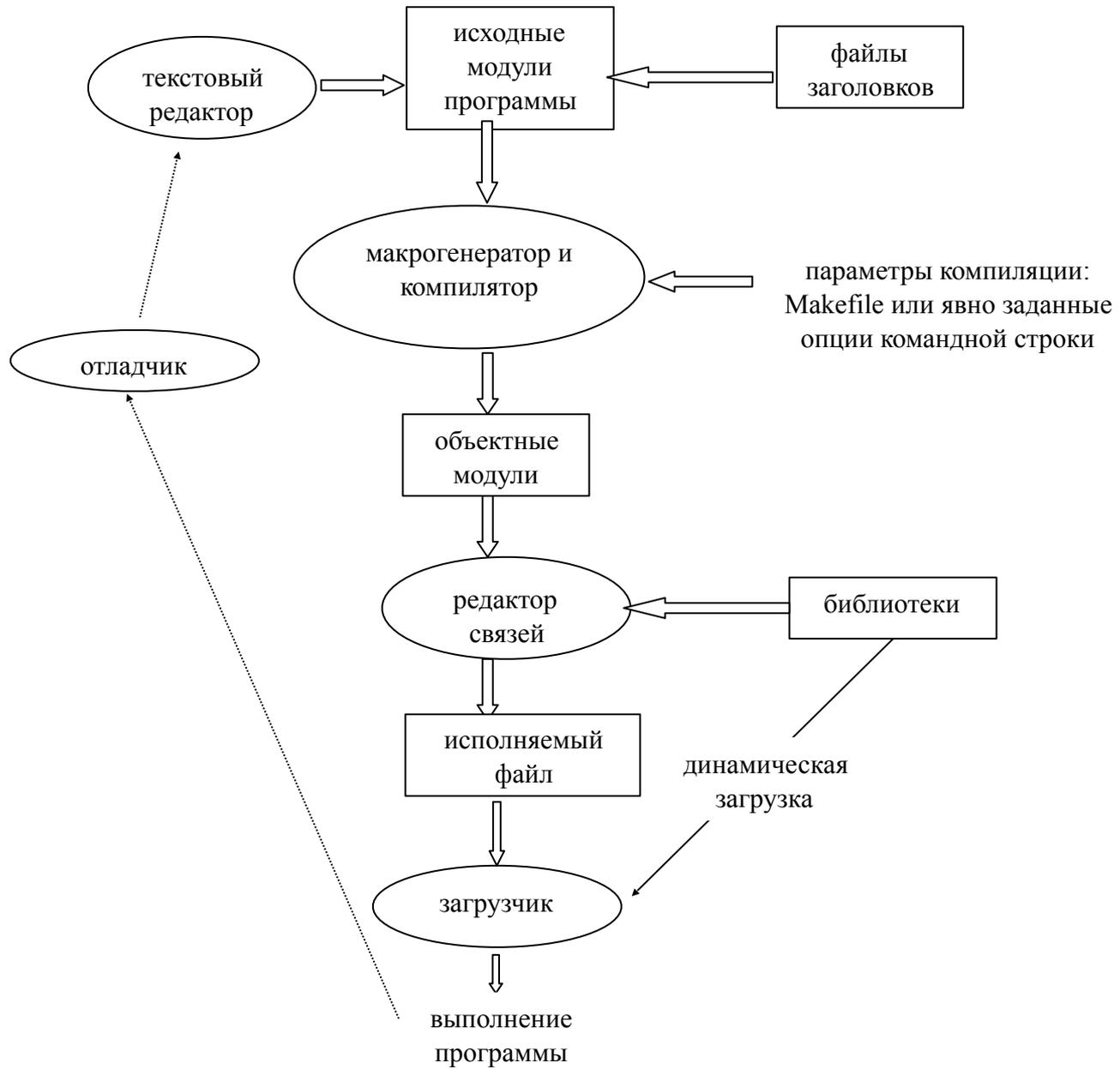
- Интегрированные  
Microsoft Visual Studio + Team Foundation
- «Кусочные»  
gcc + vim + make + gdb + git + gerrit

# Серебряной пули нет

- Инструмент для задачи
- Профессионал умеет работать со всем
- СП – не личный инструмент
  - командная работа
  - унификация

# Основные компоненты

- Редактор
- Транслятор  
компилятор+компоновщик, интерпретатор
- Стандартная библиотека
- Система сборки
- Система контроля версий
- Отладчик



C



# Python

# Дополнительные компоненты

- Инструменты визуальной разработки
- Инструменты статического анализа кода
- Профилировщик
- Средства тестирования
- Инструменты обратного конструирования

# Дополнительные компоненты

- Система управления задачами (тикетинг, багтрекер)
- Система рецензирования кода
- Средства документирования (код, вики)
- Инструменты унификации стиля кодирования

# Два слова об IDE

- IDE (integrated development enviroment)  
ИСП (интегрированная среда разработки)
- Локальная часть СП с высокой степенью интеграции:
  - редактор – отладчик
  - редактор – средства статического анализа
  - ...

# Текстовые редакторы

# Функции текстового редактора

- Написание/редактирование программного кода
- Работа с несколькими файлами
- Подсветка синтаксиса
- Интеграция с системой сборки
- Интеграция с отладчиком

# Функции текстового редактора

- Интеграция со средствами статического анализа кода:
  - автодополнение, справка по вызовам
  - навигация
  - рефакторинг

```
import string
print string.split()
```

split(s [,sep [,maxsplit]]) -> list of strings  
Return a list of the words in the string s, using sep as the  
delimiter string.  
can be inserted by typing the snippet name followed by

```
NSLog(@"##### CALL OUT ACCESSORY TAPPED: control button type = %d", (C
```

```
if (((UIButton *)control).buttonType == 2) {
```

```
NSLog(@"##### UIButton * control tapped: in 0x176970 type = disclosure"  
leftCallOutButton.available
```

UIControl	UIControl	{...}
NSMutableDictionaryRef	_contentLookup	0x176bb0
UIEdgeInsets	_contentEdgeInsets	{...}
UIEdgeInsets	_titleEdgeInsets	{...}
UIEdgeInsets	_imageEdgeInsets	{...}
UIImageView *	_backgroundView	0x0
UIImageView *	_imageView	0x196aa0
UILabel *	_titleLabel	0x0
struct {...}	_buttonFlags	{...}
unsigned int	reversesTitleShadowWhenHighlighted	0
unsigned int	adjustsImageWhenHighlighted	1
unsigned int	adjustsImageWhenDisabled	1
unsigned int	autosizeToFit	0
unsigned int	disabledDimsImage	0
unsigned int	showsTouchWhenHighlighted	0
unsigned int	buttonType	2
unsigned int	shouldHandleScrollerMouseEvent	1

GDB: Stopped after step

```
import string
print string.s
```

- rsplit
- rstrip
- split
- splitfields
- strip

**Библиотеки**

# Библиотеки

- Весь код написать невозможно
- Все задачи решать не нужно

# Виды задач

- *Стандартные общезначимые*  
Сортировка массива, организация списка;  
работа с файлами, с популярными протоколами
- *Стандартные специализированные*  
Сложная математика, специализированные  
форматы и протоколы
- *Уникальные для продукта*

# Виды задач

- *Стандартные общезначимые*  
**В стандартной библиотеке СП**
- *Стандартные специализированные*  
**В других библиотеках**
- *Уникальные для продукта*  
**Эти задачи нужно решать**

# Зависимости

- Цена использования библиотеки – создание *зависимости* от нее
- Зависимость от стандартной библиотеки – как правило, не проблема

# Зависимости

- Увеличивают хрупкость кода
- Создают риски: каково качество кода в библиотеках?
- Большое количество зависимостей – дорога в dependency hell

TECHNOLOGY LAB —

# Rage-quit: Coder unpublished 17 lines of JavaScript and “broke the Internet”

Dispute over module name in npm registry became giant headache for developers.

SEAN GALLAGHER - 3/25/2016, 5:10 AM

## The npm Blog

Blog about npm things.



### Incident report: npm, Inc. operations incident of January 6, 2018

On Saturday, January 6, 2018, we incorrectly removed the user `floatdrop` and blocked the discovery and download of all 102 of their packages on the public npm Registry. Some of those packages were highly depended on, such as `require-from-string`, and removal disrupted many users' installations.

On Sunday, we published [an initial blog post](#) to clarify that this issue was an internal operations issue and not a security issue, but at the time of that post we lacked many details because we had not yet conducted npm's post-incident retrospective process. This disclosure follows our retrospective and goes into detail about how this mistake happened and what actions we've already taken and will take to prevent similar incidents.

A full list of the affected packages is at the end of this post.



# Зависимости

- Сцилла: изобретение велосипеда
- Харибда: 100500 зависимостей и 1 строка кода

# Системы сборки

# Системы сборки

- Задача: автоматизированно построить по исходного кода ПП артефакты, пригодные для конечного пользователя
  - исполняемые файлы, дистрибутивный комплект, ...
- Исключить рутину и возможные ошибки
- Не собирать то, что уже собрано
- + Запуск тестов и проч.

# Примеры

- `.c`, `.h`  $\rightarrow$  `.o`  $\rightarrow$  исполняемый файл
- Веб-фронтенд (`.js`, `.css`)
- Конкретные системы:
  - `make`, `msbuild`, `ant`, `Rake`, `webpack`, `Gulp`, ...

# Системы контроля версий

# Системы контроля версий

- Хранят версии (ревизии) исходного кода и других ресурсов ПП
- Делают возможной совместную работу:
  - Изменение одной и той же области одновременно
  - Поддержка жизненного цикла изменения
- Позволяют «вернуться в прошлое»

# Исторический экскурс

- SCCS (1972), RCS (1982)  
*работа с отдельными файлами*
- CVS (1990), SVN (2000)  
*работа с проектом*
- BitKeeper (1998), Darcs (2002),  
Git (2005), Mercurial (2005)  
*распределенные системы*

# Основные понятия

## *Сущности*

Дерево

Ревизия

Набор изменений  
(changeset)

Ветка

Репозиторий

Рабочая копия

## *Операции*

Фиксация

(commit)

Обновление на ревизию

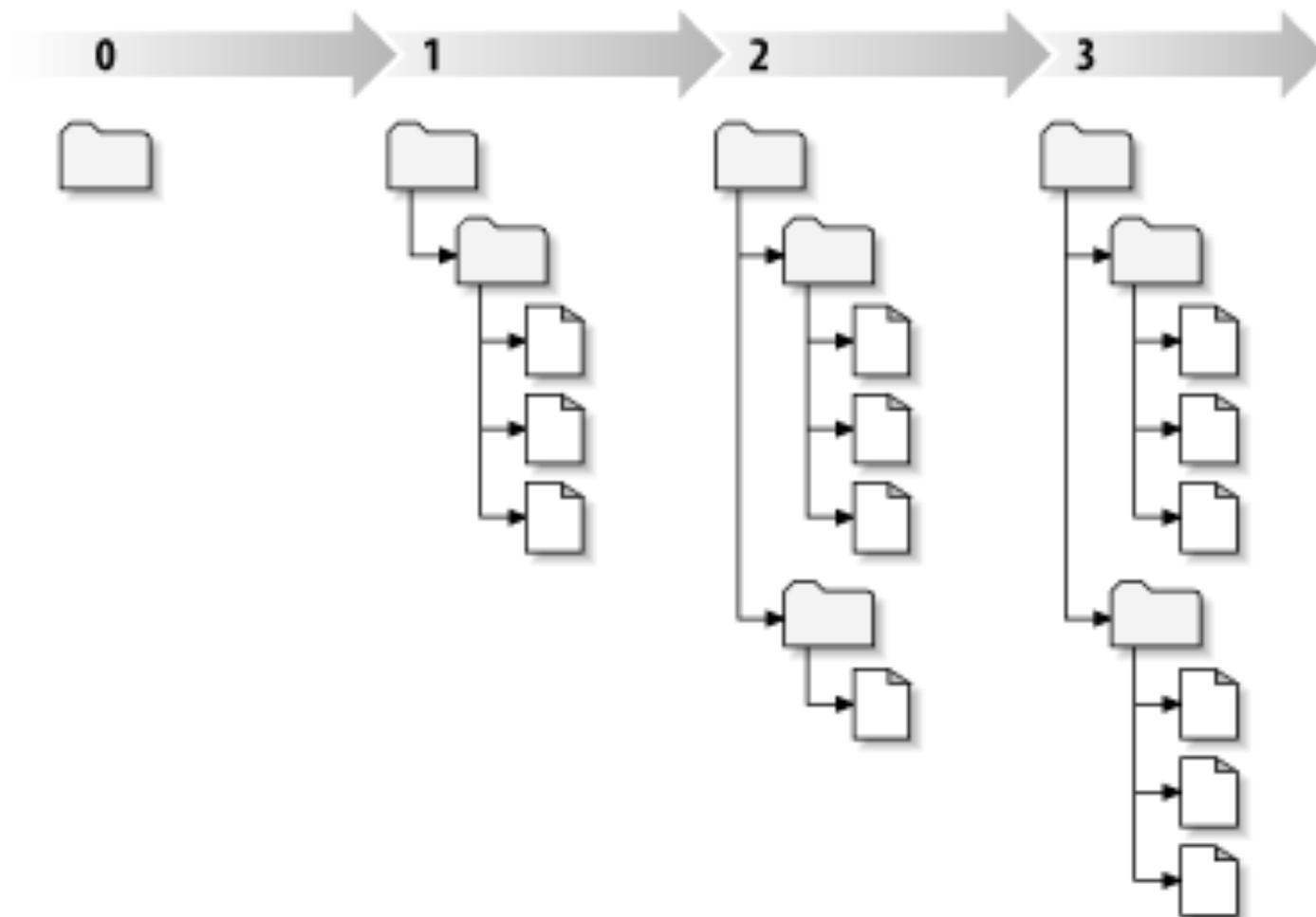
Ветвление

Слияние

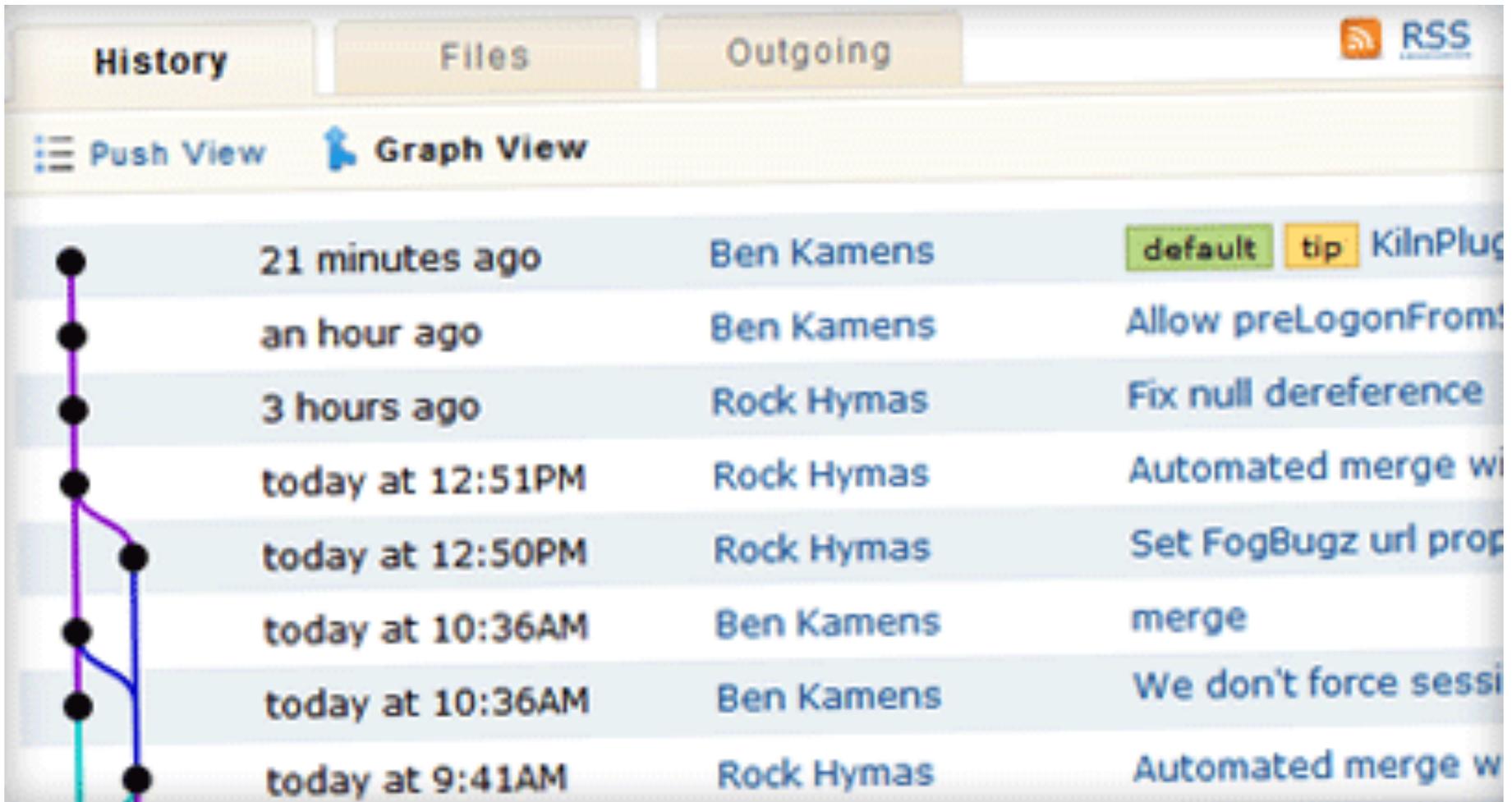
(merge)

Передача изменений  
(pull/push)

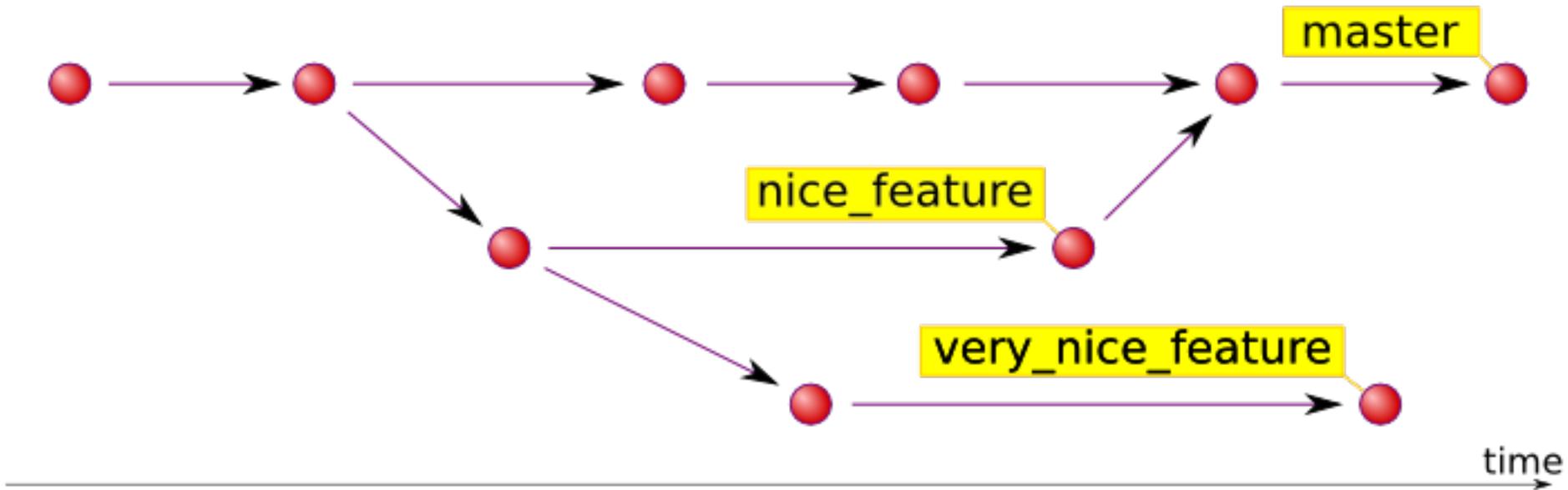
# Изменение дерева файлов



# Изменение дерева файлов



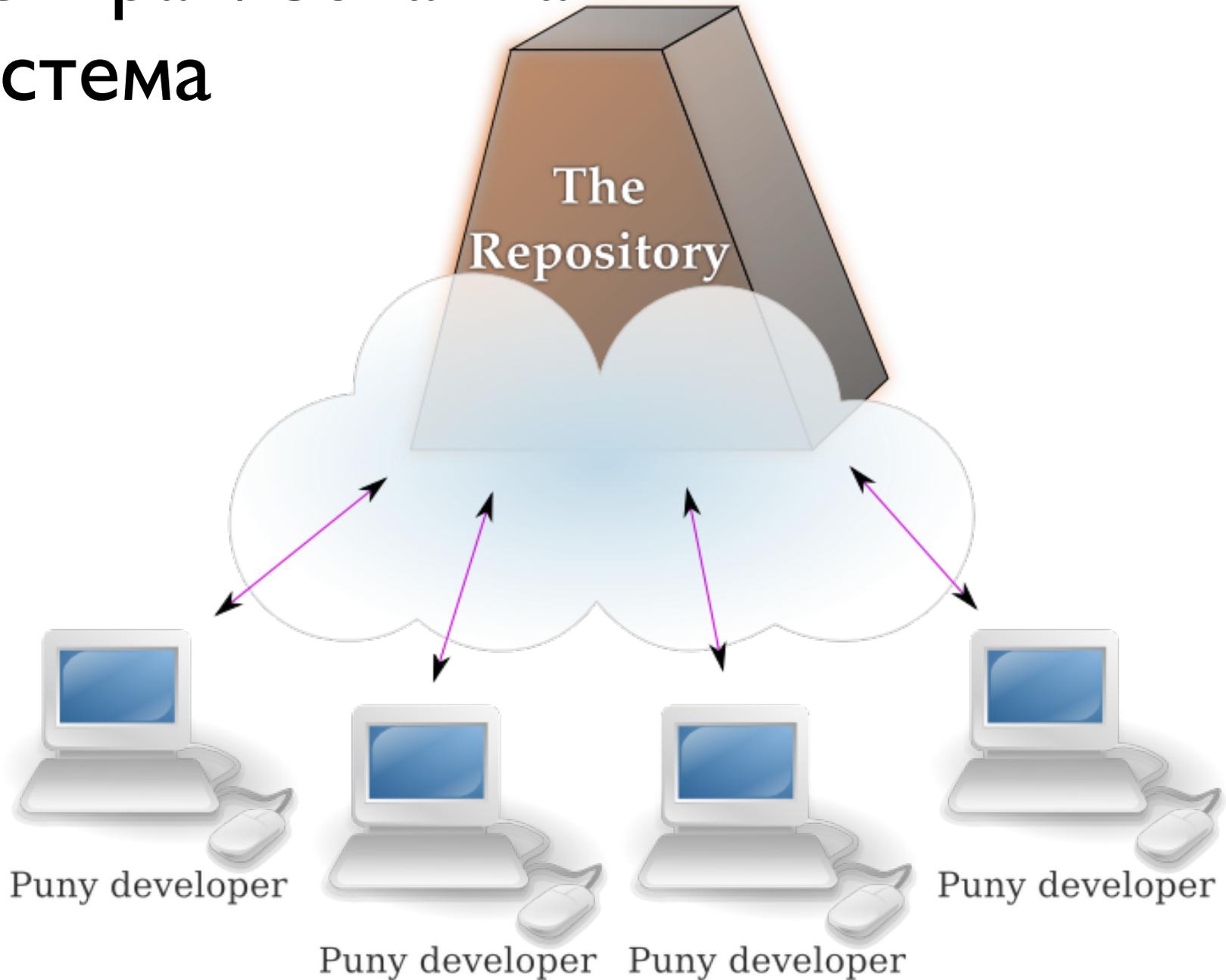
# Ветки



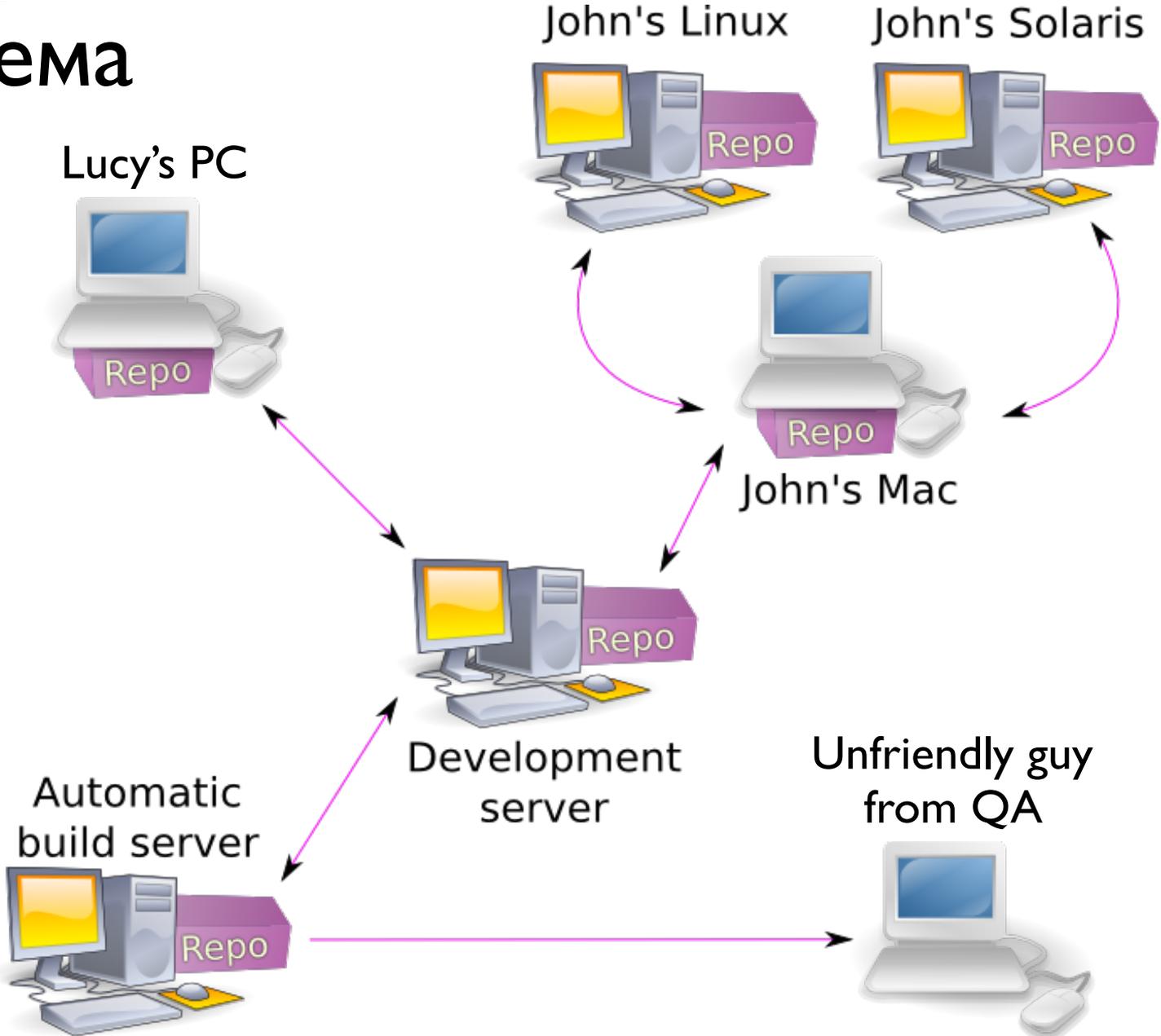
# Системы контроля версий

- Централизованные
  - Контроль
  - Борьба с избыточностью
- Распределенные
  - Автономность
  - Резервирование

# Централизованная система



# Распределенная система



# Приемы работы

- Линейная работа
- Ветка на задачу
- Ветка для стабилизации
- Ветка для сопровождения версии
- Метки (тэги) для версий
- Анализ истории

# Аннотирование кода

	109	
[1586]	110	<code>class ReportModule(Component):</code>
[1]	111	
[1860]	112	<code>implements(INavigationContributor, IPermissionRequestor, IRequestHandler,</code>
	113	<code>IWikiSyntaxProvider)</code>
[1586]	114	
[6901]	115	<code>items_per_page = IntOption('report', 'items_per_page', 100,</code>
	116	<code>    """Number of tickets displayed per page in ticket reports,</code>
	117	<code>    by default ('since 0.11')""")</code>
	118	
	119	<code>items_per_page_rss = IntOption('report', 'items_per_page_rss', 0,</code>
	120	<code>    """Number of tickets displayed in the rss feeds for reports</code>
	121	<code>    ('since 0.11')""")</code>
[11096]	122	
[1586]	123	<code># INavigationContributor methods</code>
	124	
	125	<code>def get_active_navigation_item(self, req):</code>
	126	<code>    return 'tickets'</code>
	127	
	128	<code>def get_navigation_items(self, req):</code>
[4143]	129	<code>    if 'REPORT_VIEW' in req.perm:</code>
[5776]	130	<code>        yield ('mainnav', 'tickets', tag.a(_('View Tickets'),</code>
[4787]	131	<code>            href=req.href.report()))</code>
[1586]	132	
[11096]	133	<code># IPermissionRequestor methods</code>
[1860]	134	
[11096]	135	<code>def get_permission_actions(self):</code>



# Какую систему использовать?

- Git  
(если у вас возник этот вопрос)
- Если в команде принята какая-то система, то ее – вопроса нет
- Если вы знаете, почему вам не подходит Git – вопроса тоже нет

**Когда система контроля  
версий нужна, а когда нет?**

Когда система контроля  
версий нужна, а когда нет?

Нужна всегда\*

---

\* За исключением однострочников

**За рамками этой лекции**

- Непрерывная интеграция
- Ревью кода
- Хранилище артефактов
- DevOps
- Трансляторы для кросс-платформенной разработки (Xamarin)
- CASE-средства
- Средства анализа кода (lint)
- Программная археология

# Q & A

слайды:

[warmland.ru/cs/sp/](http://warmland.ru/cs/sp/)

обратная связь:

<https://forms.gle/RLAhBK5BXJuugVGq6>

вопросы:

[franoleg@gmail.com](mailto:franoleg@gmail.com)