Системы программирования. Системы контроля версий. Жизненный цикл ПО

Олег Французов
2019

Интро
Цель: понимать, как устроен проект по разработке ПО
потом — вопросы и ответы

Что такое *система программирования?*

Вы пользовались системой программирования?

Структура вычислительной системы

Компьютер всегда существует ради прикладного ПО, будь это телефон или супер-ЭВМ

Что такое
программный продукт?

Практикум – ПП?

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.

на каждом этапе, на каждом шаге требуется инструментальная поддержка; эти инструменты разнородны, но они и составляют СП

Жизненный цикл программного продукта

Давайте поговорим о ЖЦ
Будем говорить о коммерческой разработке

Жизненный цикл ПП

- Разработка
- Внедрение
- Сопровождение

Software life cycle management: development, installation/deployment, maintenance
(В конце – вывод из эксплуатации, disposal)

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

Анализа требований в заданиях практикума не бывает
Это не последовательные шаги
«Фрактал»

- 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*

В случае коробочного продукта заказчик – менеджер продукта, который знает рынок сбыта

Анализа требований в заданиях практикума не бывает!

Анализ требований

как он бывает на самом деле

(*видео*)

The Expert (Short Comedy Sketch)
https://www.youtube.com/watch?v=BKorP55Aqvg

D. Scott Williamson, Expert
https://www.youtube.com/watch?v=B7MlJP90biM

Проектирование

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

Фрактальная структура
Создание продукта, системы, подсистемы, модуля

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

Как мало места и времени занимает кодирование
10 lines of code per day

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

Самый непредсказуемый шаг

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?

Ручное и автоматизированное
Регрессионное

Виды тестирования

- Тестирование черного ящика
- Тестирование белого ящика

ЧЯ - функциональное тестирование
БЯ - модульное тестирование

Документирование

- Пользовательская документация
- Документация для разработчиков

Модели жизненного цикла разработки

- Каскадная (водопад)
- Итерационная (agile)

история вопроса

(*видео*)

The Rise and Fall of Waterfall
http://vimeo.com/18951935

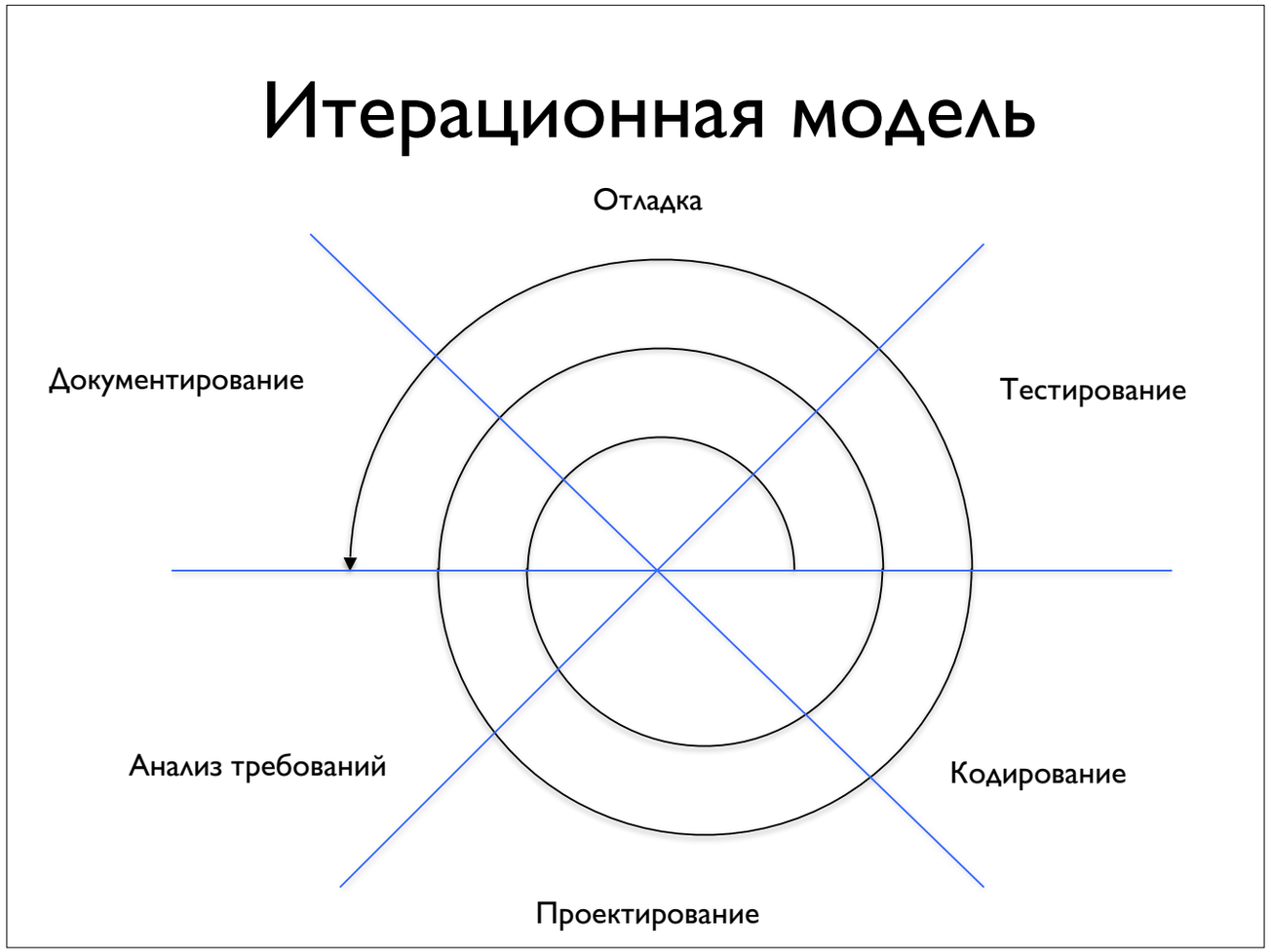
Каскадная модель

Водопад
Любое отклонение – катастрофа
Фредерик Брукс. Мифический человеко-месяц

Каскадно-возвратная модель

Возврат с конца – катастрофа

Итерационная модель



Agile (SCRUM)
Каждая итерация (спринт) – фиксированное время
Невозможно потерять слишком много времени

Системы программирования

По стратегии трансляции

- Компиляция
- Интерпретация
- Смешанная стратегия

JIT-компиляция, промежуточный код

Примеры

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

JIT-компиляция, промежуточный код

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

Это спектр, а не «или-или», чистого нет
Чем типичнее задача в «потоке», тем полезнее первые
Разработка сайтов vs. ядро ОС

Примеры

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

Крупные компании - частные решения

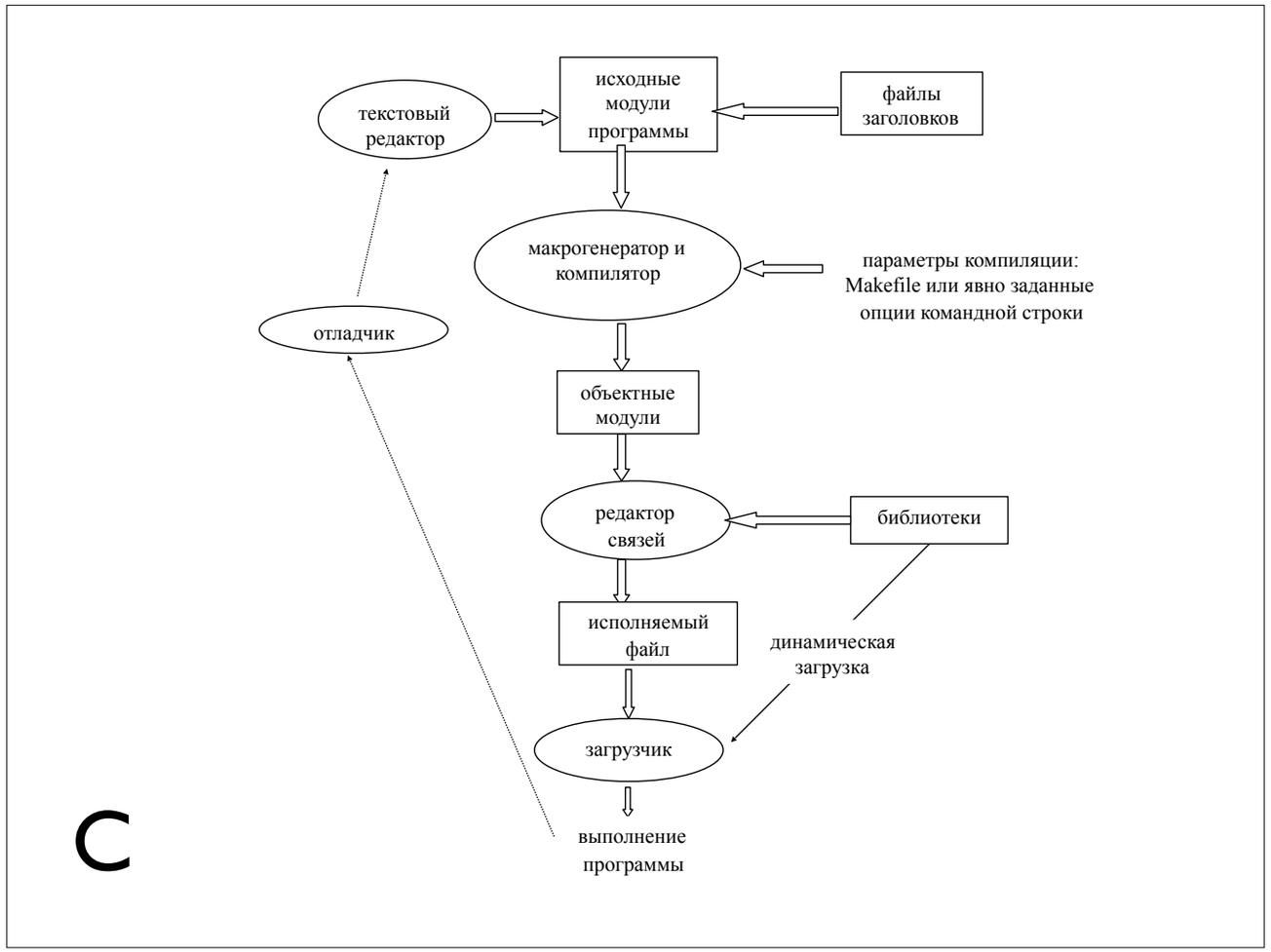
Серебряной пули нет

- 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

Примеры из разных областей


текстовый редактор
исходные модули программы
файлы заголовков
макрогенератор и компилятор
параметры компиляции: Makefile или явно заданные опции командной строки
отладчик
объектные модули
редактор связей
библиотеки
исполняемый файл
динамическая загрузка
загрузчик
выполнение программы

C


текстовый редактор
исходные модули программы
интерпретатор
отладчик
байт-код
Среда времени выполнения
интерпретатор
библиотеки
выполнение программы

Python

Дополнительные компоненты

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

Личные/локальные
Примеры из разных областей

Дополнительные компоненты

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

Не личные / нелокальные
Примеры из разных областей

Два слова об IDE

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

Проблематика «не здесь»
Далее мы не будем разграничивать IDE и не-IDE
Поговорим подробнее о важнейших частях СП

Текстовые редакторы

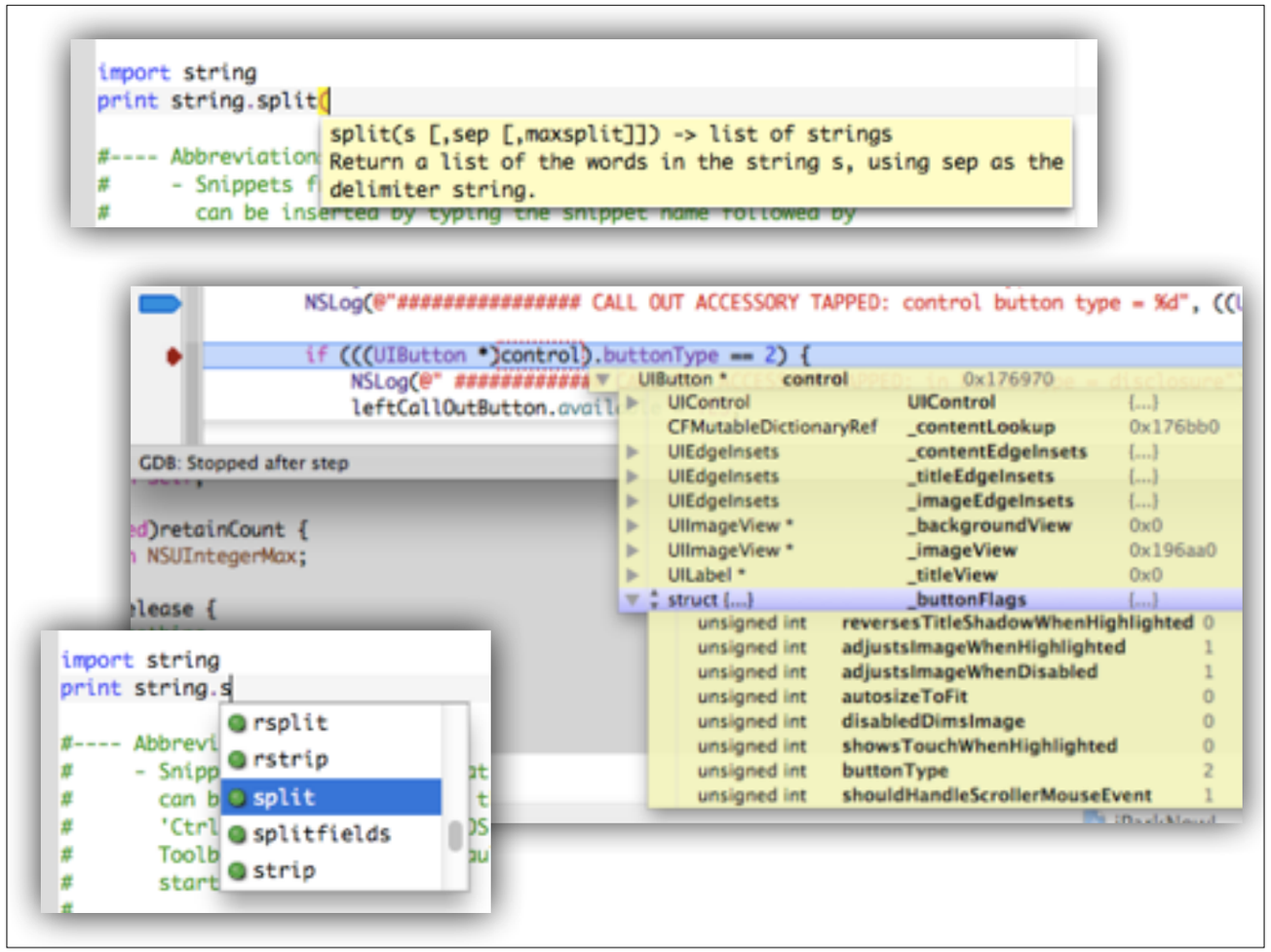
Функции текстового редактора

- 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

Рефакторинг — формальное преобразование кода, не приводящее к изменению поведения программы


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.
NSLog(@"################ CALL OUT ACCESSORY TAPPED: control button type = %d", ((
if (((UIButton *)control).buttonType == 2) {
NSLog(@"
leftCallOutButton.
UIButton * control 0x176970
GDB: Stopped after step
UIControl UIControl [...]
CFMutableDictionaryRef _contentLookup 0x176bb0
UIEdgeInsets _contentEdgeInsets [...]
UIEdgeInsets _titleEdgeInsets [...]
UIEdgeInsets _imageEdgeInsets [...]
UIImageView * _backgroundView 0x0
UIImageView * _imageView 0x196aa0
UILabel * _titleView 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
)retainCount {
NSUIntegerMax;
lease {
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 самостоятельно

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

проблемы Dependency Hell на примере Node.js
2016 – модуль left-pad
2018 - missing packages incident
http://arstechnica.com/information-technology/2016/03/rage-quit-coder-unpublished-17-lines-of-javascript-and-broke-the-internet/
http://blog.npmjs.org/post/169582189317/incident-report-npm-inc-operations-incident-of

Зависимости

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

Возможные критерии включения библиотеки

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 -> .o ->` 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

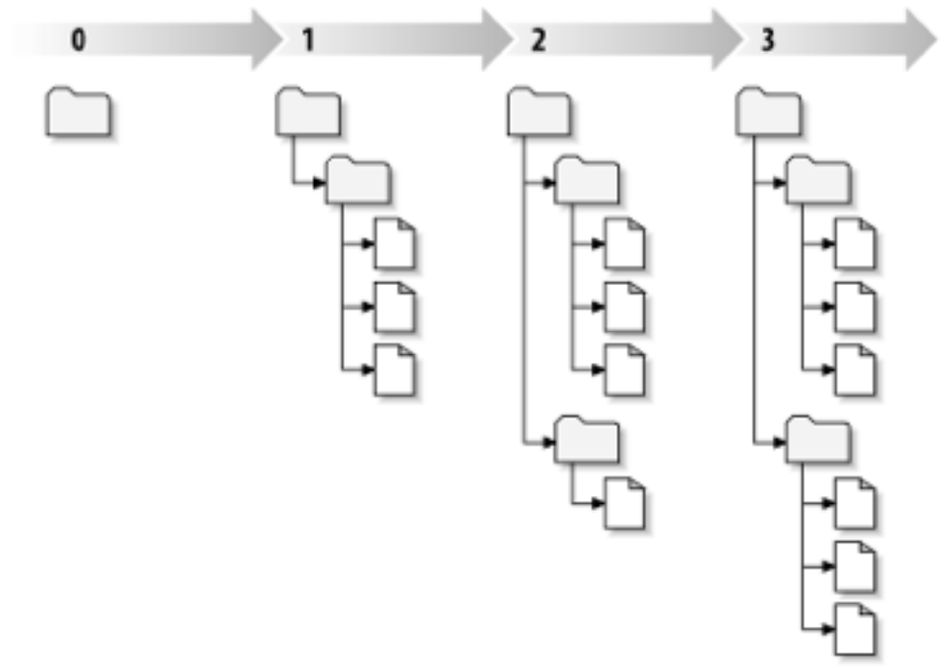
Дерево
Ревизия
Набор изменений (changeset)
Ветка
Репозиторий
Рабочая копия

Операции

Фиксация (commit)
Обновление на ревизию
Ветвление
Слияние (merge)
Передача изменений (pull/push)

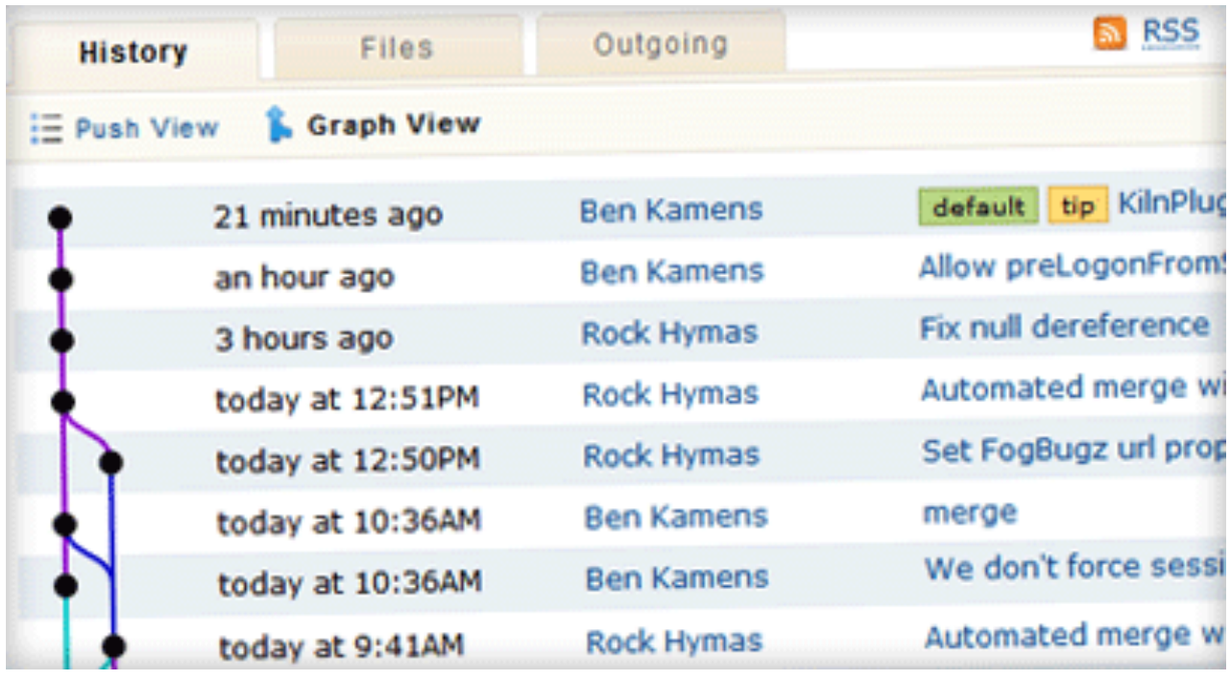
Набор изменений – разница между двумя ревизиями. Если r и p – соседние ревизии, то набор изменений A – это такое изменение ревизии r, что после его применения получается ревизия p: p = Ar.

Изменение дерева файлов



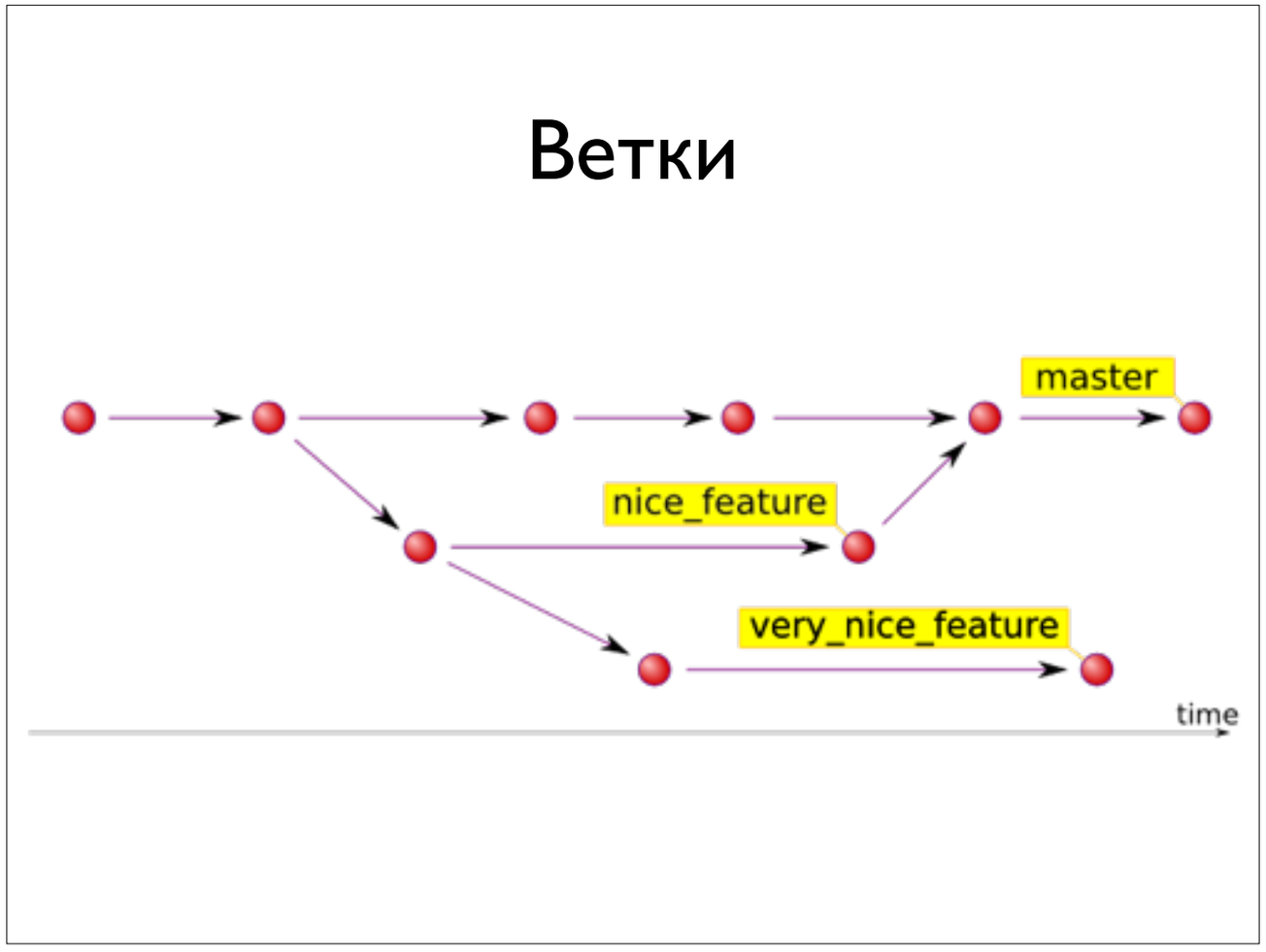
Комментарий: временное измерение, добавленное к дереву файлов. Изображены ревизии 0-3.
Объяснить подробно, где дерево, ревизия, набор изменений.

Изменение дерева файлов



Не показаны состояния дерева, но показаны наборы изменений и их связь. Видны ветки.

Ветки

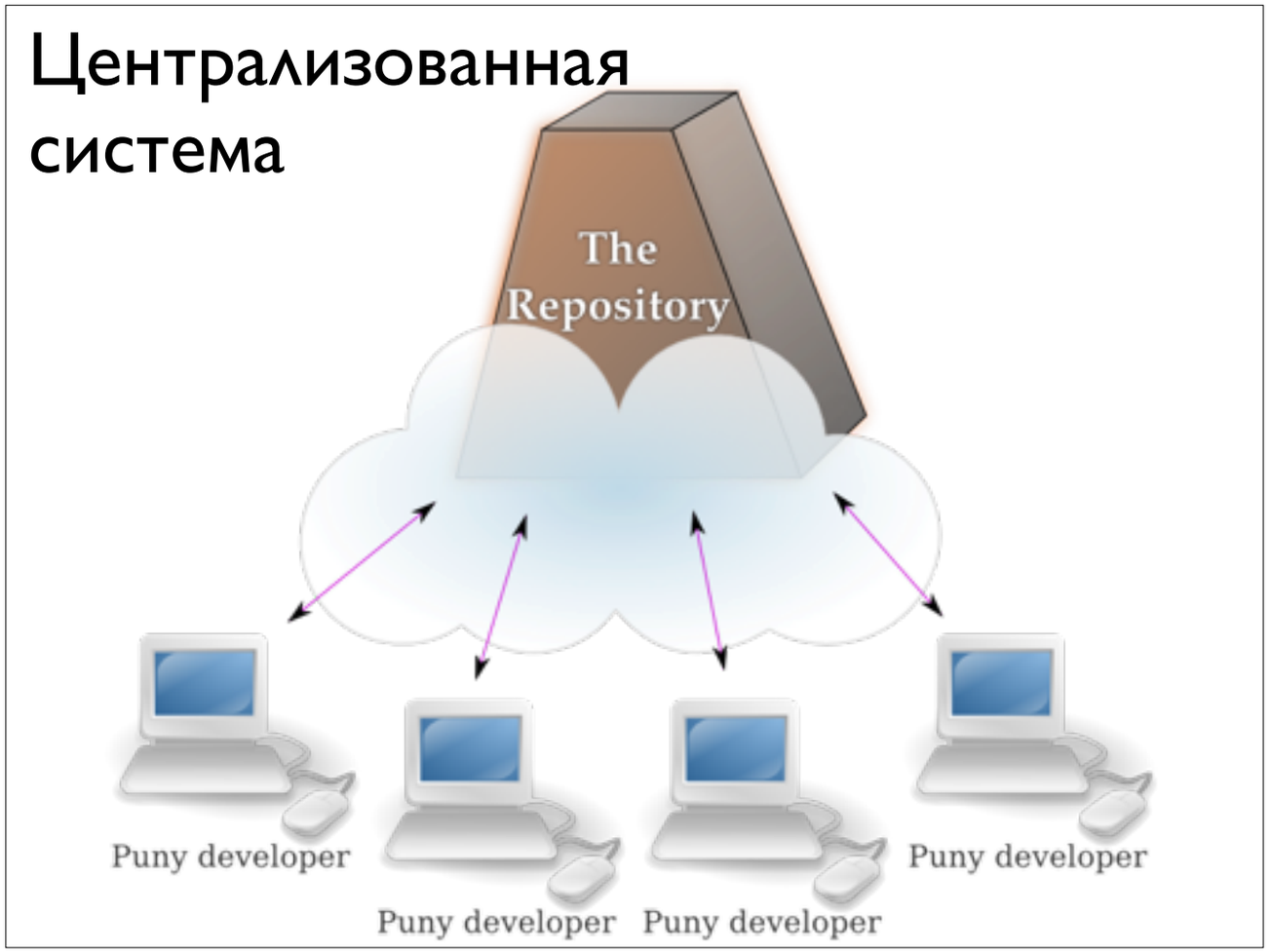


Для параллельной разработки

Системы контроля версий

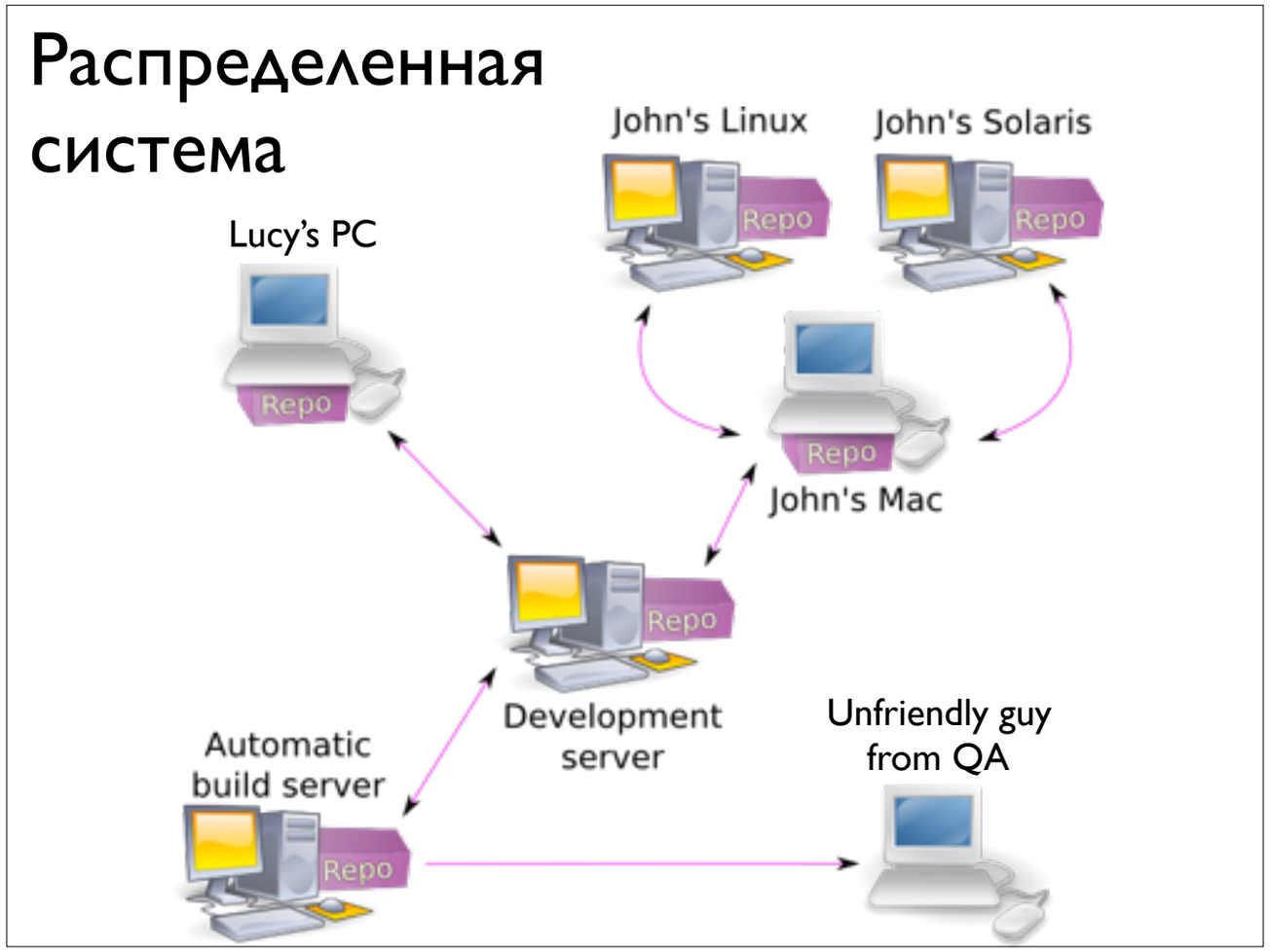
- Централизованные
  - Контроль
  - Борьба с избыточностью
- Распределенные
  - Автономность
  - Резервирование

Централизованная система



Величественный репозиторий и ничтожные разработчики. У каждого только рабочая копия. Для любых операций, от фиксации до просмотра истории, требуется обращение к репозиторию.

Распределенная система



Рабочая копия и репозиторий есть на каждой машине (кроме тестировщика-QA). Все операции можно делать локально. Обмен изменениями – явная операция.

Приемы работы

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

Аннотирование кода

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

Слева номера ревизий. Старые строчки холодным цветом, новые теплым. По щелчку на ревизию можно получить информацию о наборе изменений.

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

Если вам говорят, что нам не нужна СКВ – бегите!

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/

вопросы:
franoleg@gmail.com