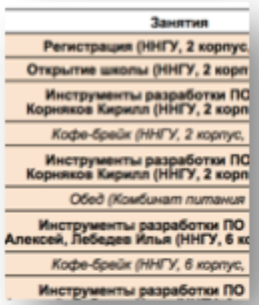
Инструменты разработки ПО

Кирилл Корняков
Директор по исследованиям и разработке, Itseez
Февраль 2016

Скелетонизация


Занятия
Регистрация (ННГУ, 2 корпус,
Открытие школы (ННГУ, 2 корп
Инструменты разработки ПО
Корняков Кирилл (ННГУ, 2 корп
Кофе-брейк (ННГУ, 2 корпус,
Инструменты разработки П
Корняков Кирилл (ННГУ, 2 корп
Обед (Комбинат питания
Инструменты разработки ПО
Алексей, Лебедев Илья (ННГУ, 6 ко
Кофе-брейк (ННГУ, 6 корпус,
Инструменты разработки ПО

Используемые инструменты


Занятия
Регистрация (ННГУ, 2 корпус
Открытие школы (ННГУ, 2 корп
Инструменты разработки ПО
Корняков Кирилл (ННГУ, 2 корп
Кофе-брейк (ННГУ, 2 корпус,
Инструменты разработки ПО
Корняков Кирилл (ННГУ, 2 корп
Обед (Комбинат питания
Инструменты разработки ПО
Алексей, Лебедев Илья (ННГУ, 6 к
Кофе-брейк (ННГУ, 6 корпус,
Инструменты разработки ПО


OpenCV


CMake
Cross-platform Make


gi

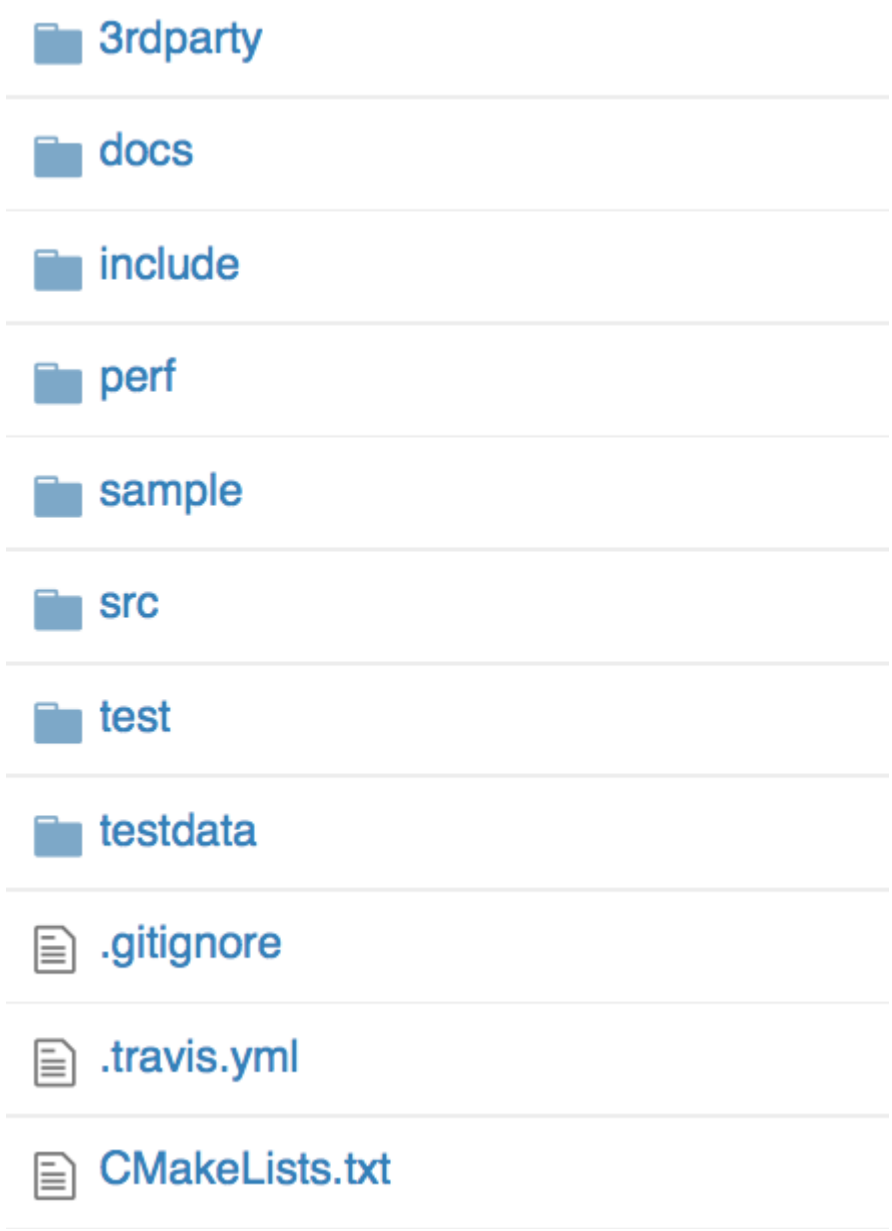

googletest
Google C++ Testing Framework

Gitl

Содержание

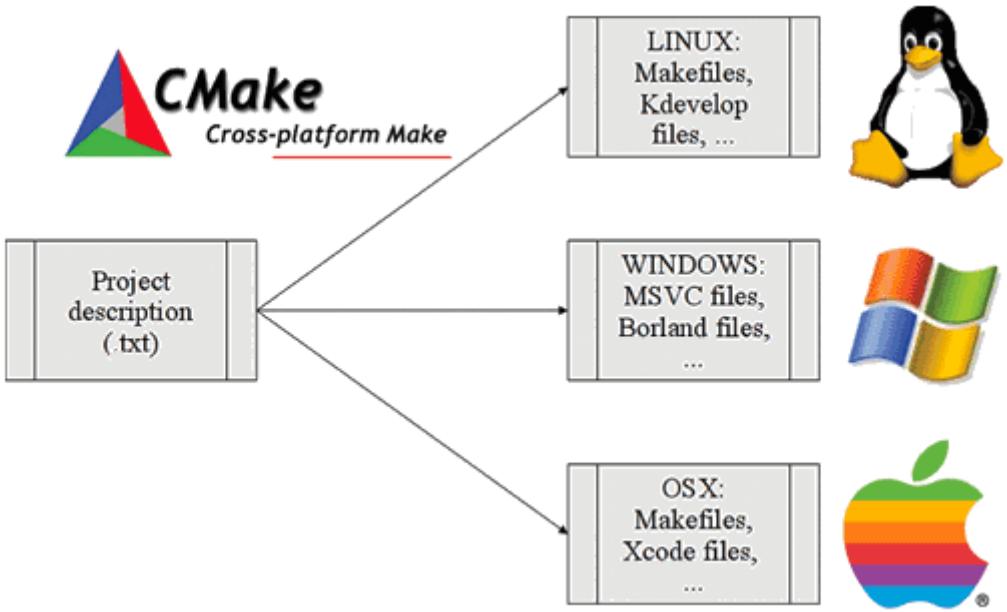
1. Кросс-платформенная разработка
    - *[C++], CMake*
2. Коллективная работа с кодом
    - *Git, GitHub*
3. Автоматическое тестирование
    - *Google Test, Travis-CI*

Программная реализация


3rdparty
docs
include
perf
sample
src
test
testdata
.gitignore
.travis.yml
CMakeLists.txt


CMake
Cross-platform Make

CMake

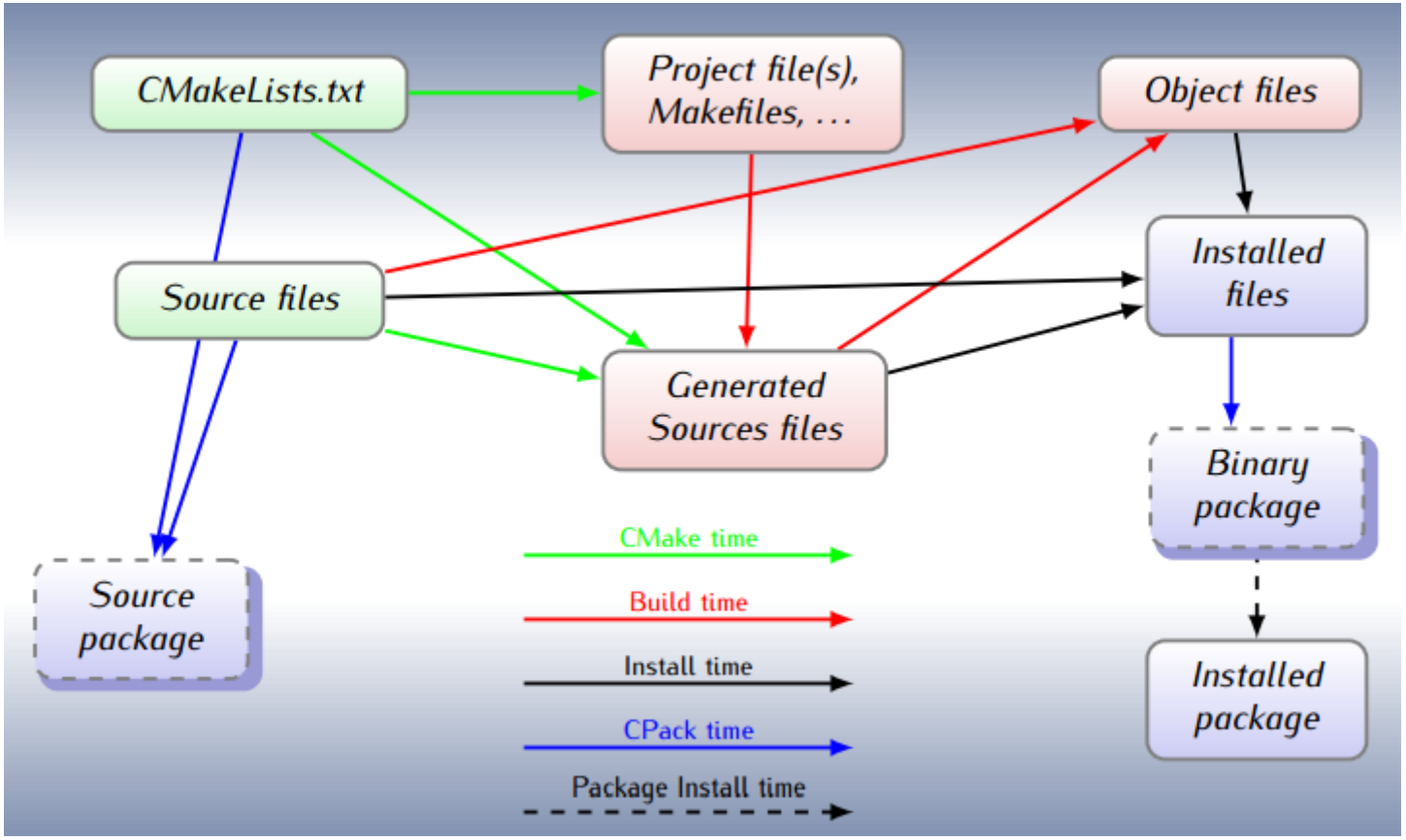


- Широкая поддержка разнообразных целевых платформ и IDE

- Максимальная свобода в выборе окружения разработки (в рамках одной команды!)
- В настоящий момент является стандартом де-факто для C++ проектов

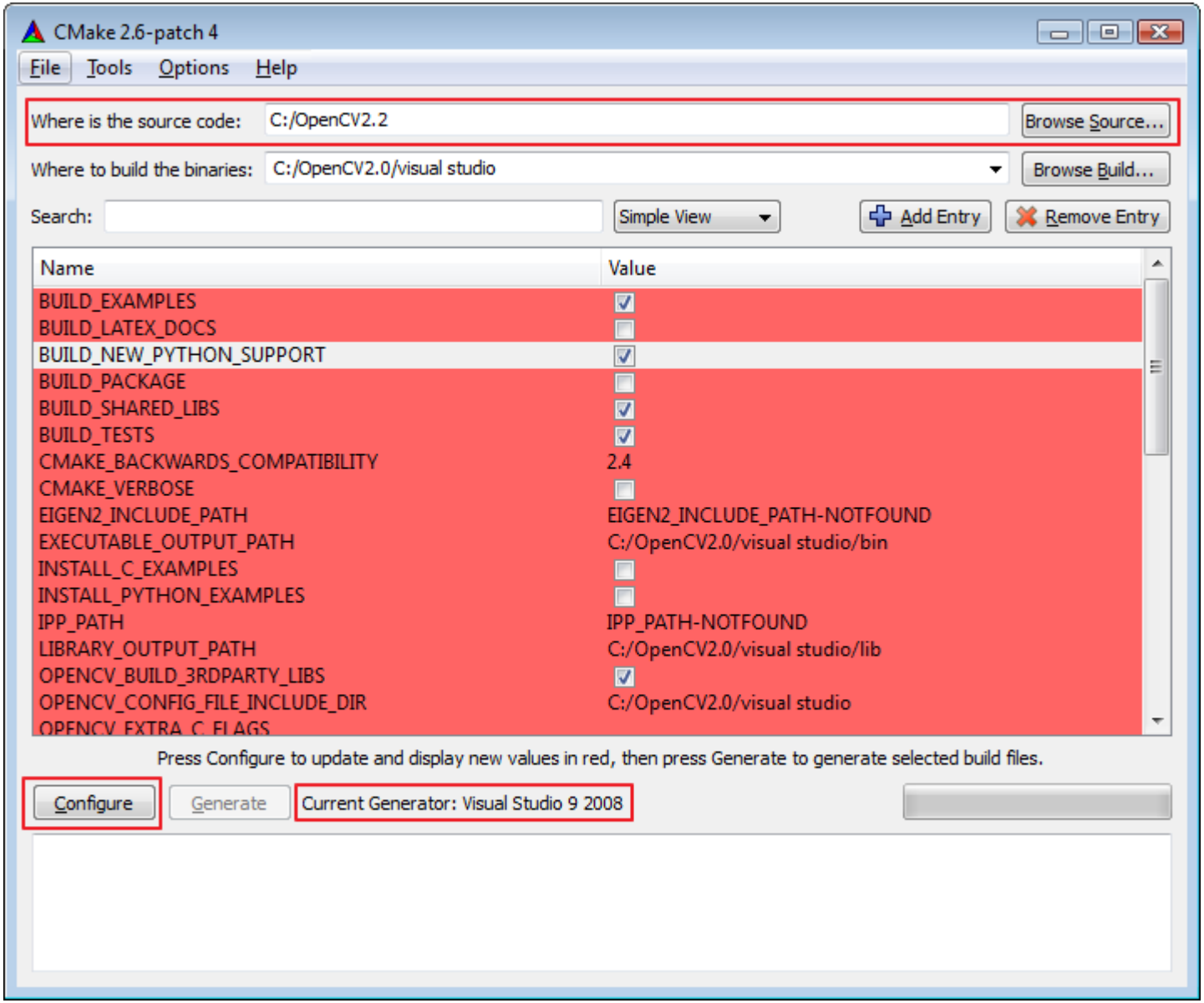
CMake Workflow

`CMakeLists.txt` — файл, описывающий порядок сборки приложения



- **Шаг 0.** помощи
  - *.vcp*
- **Шаг 1.** помощи Creator,
  - *.obj*
- **Шаг 2.** файлов
  - *.exe*

CMake GUI


CMake 2.6-patch 4
File Tools Options Help
Where is the source code: C:/OpenCV2.2
Browse Source...
Where to build the binaries: C:/OpenCV2.0/visual studio
Browse Build...
Search:
Simple View
Add Entry
Remove Entry
Name
Value
BUILD_EXAMPLES
BUILD_LATEX_DOCS
BUILD_NEW_PYTHON_SUPPORT
BUILD_PACKAGE
BUILD_SHARED_LIBS
BUILD_TESTS
CMAKE_BACKWARDS_COMPATIBILITY 2.4
CMAKE_VERBOSE
EIGEN2_INCLUDE_PATH EIGEN2_INCLUDE_PATH-NOTFOUND
EXECUTABLE_OUTPUT_PATH C:/OpenCV2.0/visual studio/bin
INSTALL_C_EXAMPLES
INSTALL_PYTHON_EXAMPLES
IPP_PATH IPP_PATH-NOTFOUND
LIBRARY_OUTPUT_PATH C:/OpenCV2.0/visual studio/lib
OPENCV_BUILD_3RDPARTY_LIBS
OPENCV_CONFIG_FILE_INCLUDE_DIR C:/OpenCV2.0/visual studio
Press Configure to update and display new values in red, then press Generate to generate selected build files.
Configure
Generate
Current Generator: Visual Studio 9 2008

Пример сборки приложения (add_executable)

Содержимое каталога:

```
code
├── CMakeLists.txt
├── lib.h
├── lib.c
└── main.c
```

CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8)
project(first_sample)

set(SOURCES main.c lib.c)
add_executable(sample_app ${SOURCES}) # Объявляет исполняемый модуль с именем sam
```

Out of source build

Плохо: в директории с исходным кодом

```
code
├── hello.hpp
├── hello.cpp
└── hello.exe # Этот файл может случайно попасть в историю Git
```

Хорошо: вне директории (чистый репозиторий, несколько build-директорий)

```
code
├── hello.hpp
└── hello.cpp
build
└── hello.exe
```

Соответствующие команды:

```
$ cd <code>
$ mkdir ../build
$ cd ../build
$ cmake ../code
$ make
```

Пример сборки библиотеки (add_library)

Содержимое каталога:

```
code
├── CMakeLists.txt
├── lib.h
├── lib.c
└── main.c
```

CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8)
project(second_sample)

set(SOURCE_LIB lib.c)
add_library(library STATIC ${SOURCE_LIB}) # Объявляет библиотеку с именем library
```

```
set(SOURCES main.c)
add_executable(main ${SOURCES}) # Объявляет исполняемый модуль с именем sample_ap
target_link_libraries(sample_app library) # Указывает зависимость от библиотеки
```

Добавление подпроекта

Содержимое каталога:

```
code
├── CMakeLists.txt
├── library
│   ├── CMakeLists.txt
│   ├── lib.c
│   └── lib.h
└── main.c
```

Корневой `CMakeLists.txt`:

```
cmake_minimum_required(VERSION 2.8)
project(third_sample)

add_subdirectory(library) # Указывает, что в директории library есть свой CMakeLi

include_directories(library)
```

```
set(SOURCES main.c)
add_executable(sample_app ${SOURCES})

target_link_libraries(sample_app library)
```

library/CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8)
project(library)

set(SOURCE_LIB lib.c)
add_library(library STATIC ${SOURCE_LIB})
```

Поиск зависимостей

CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8)
project(sample)

# Поиск OpenCV
find_package(OPENCV REQUIRED)
if(NOT OPENCV_FOUND)
    message(SEND_ERROR "Failed to find OpenCV")
    return()
else()
    include_directories(${OPENCV_INCLUDE_DIR})
endif()

add_executable(sample_app main.c)
target_link_libraries(sample_app ${OPENCV_LIBRARIES})
```

Debug / Release

В `CMakeLists.txt`:

```
SET(CMAKE_BUILD_TYPE Debug)
```

В командной строке:

```
$ cmake -DCMAKE_BUILD_TYPE=Debug ../code # Запомните эту команду!
```

Для библиотек:

```
TARGET_LINK_LIBRARIES(lib RELEASE ${lib_SRCS})
TARGET_LINK_LIBRARIES(libd DEBUG ${lib_SRCS})
```

CMake: Резюме

- Основной "недостаток" — собственный язык
- Поначалу инструмент кажется нетривиальным, но очень удобен впоследствии
- Дает членам команды максимальную свободу в выборе инструментов
  (ОС, IDE или простой текстовый редактор)
- Обеспечивает переносимость и является стандартом де-факто
  для кросс-платформенных C++ проектов


git

Коллективная работа с кодом

1. **История изменений**
    - *Откат дефектных изменений*
    - *Извлечение кода "из прошлого" (как оно раньше работало?)*
    - *Поиск ошибок сравнением (кто это сделал?)*
2. **Централизованное хранение**
    - *Актуальное и используемое всеми участниками (где последняя версия?!)*
    - *Защищенное, с разграничением прав доступа*

Машина времени и **сетевое хранилище** в одном флаконе! Нужны ли специальные инструменты? Вспоминаем Sharepoint, tarballs.

Системы контроля версий

Системы контроля версий *— это программные системы, хранящие несколько версий одного документа, и позволяющие вернуться к более ранним версиям. Как правило, для каждого изменения запоминается дата модификации и автор.*

Патчи

Патч *(англ. patch — заплатка) — информация, предназначенная для автоматизированного внесения определённых изменений в компьютерные файлы.*

Unified diff format:

```
@@ -l,s +l,s @@ optional section heading
```

```
diff --git a/README.md b/README.md
index afadff2..bde857e 100644
--- a/README.md
+++ b/README.md
@@ -40,10 +40,10 @@ __Цель данной работы__ - реализовать набор пр
      содержащий простейшую реализацию класса матриц. Предполагается, что он не
      редактируется при реализации фильтров.
    - Модуль `filters`( `./include/filters.hpp`, `./src/filters_opencv.cpp`,
-     `./src/filters_fabrics.cpp`), содержащий объявление абстрактного класса
+     `./src/filters_factory.cpp`), содержащий объявление абстрактного класса
      фильтров (`filters.hpp`) и его наследника, который реализует перечисленные
      фильтры средствами библиотеки OpenCV (`filters_opencv.cpp`), а также метод
-     создания конкретной реализации класса фильтров (`filters_fabrics.cpp`).
+     создания конкретной реализации класса фильтров (`filters_factory.cpp`).
    - Тесты для класса матриц и фильтров (`matrix_test.cpp`, `filters_test.cpp`)
    - Пример использования фильтра (`matrix_sample.cpp`).

@@ -489,11 +489,11 @@ __Примечание:__ генератор проекта должен сов
          значение, соответствующее вашей реализации фильтра. Назовите его
          согласно вашей фамилии `YOUR_NAME`. Указанное перечисление использует
          при прогоне одних и тех же тестов на всех реализациях класса фильтров.
-     1. В файле `filters_fabrics.cpp` объявите функцию
+     1. В файле `filters_factory.cpp` объявите функцию
        `Filters* createFiltersYourName()`. Данная функция будет использована
        при создании объекта класса с вашей реализации фильтров.
      1. В функции `Filters* createFilters(FILTERS_IMPLEMENTATIONS impl)` (файл
-       `filters_fabrics.cpp`) необходимо добавить еще одну ветку у оператора
```

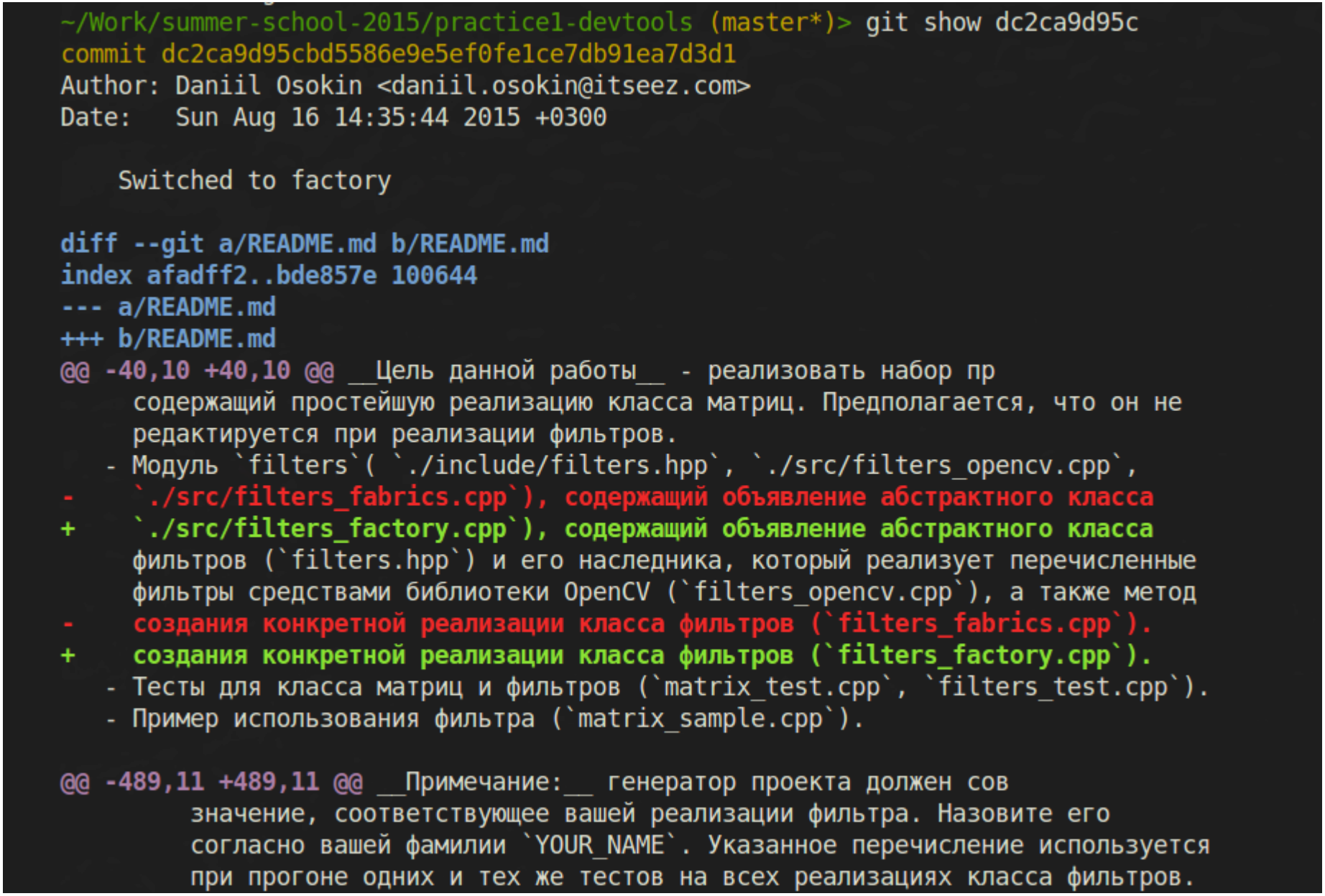
Отображение на GitHub

8 README.md

View

```
@@ -40,10 +40,10 @@ __Цель данной работы__ - реализовать набор пр
     содержащий простейшую реализацию класса матриц. Предполагается, что он не
     редактируется при реализации фильтров.
   - Модуль `filters`( `./include/filters.hpp`, `./src/filters_opencv.cpp`,
-    `./src/filters_fabrics.cpp`), содержащий объявление абстрактного класса
+    `./src/filters_factory.cpp`), содержащий объявление абстрактного класса
     фильтров (`filters.hpp`) и его наследника, который реализует перечисленные
     фильтры средствами библиотеки OpenCV (`filters_opencv.cpp`), а также метод
-    создания конкретной реализации класса фильтров (`filters_fabrics.cpp`).
+    создания конкретной реализации класса фильтров (`filters_factory.cpp`).
   - Тесты для класса матриц и фильтров (`matrix_test.cpp`, `filters_test.cpp`).
   - Пример использования фильтра (`matrix_sample.cpp`).

@@ -489,11 +489,11 @@ __Примечание:__ генератор проекта должен сов
       значение, соответствующее вашей реализации фильтра. Назовите его
       согласно вашей фамилии `YOUR_NAME`. Указанное перечисление используется
       при прогоне одних и тех же тестов на всех реализациях класса фильтров.
-    1. В файле `filters_fabrics.cpp` объявите функцию
+    1. В файле `filters_factory.cpp` объявите функцию
       `Filters* createFiltersYourName()`. Данная функция будет использована
       при создании объекта класса с вашей реализации фильтров.
     1. В функции `Filters* createFilters(FILTERS_IMPLEMENTATIONS impl)` (файл
-      `filters_fabrics.cpp`) необходимо добавить еще одну ветку у оператора-
+      `filters_factory.cpp`) необходимо добавить еще одну ветку у оператора-
       переключателя `switch`, по которой будет проходить исполнение программы,
       если создан объект класса фильтров `YOUR_NAME`.
     1. В файл `filters_YOUR_NAME.cpp` необходимо поместить реализацию функции
```

Отображение в командной строке


~/Work/summer-school-2015/practice1-devtools (master*)> git show dc2ca9d95c
commit dc2ca9d95cbd5586e9e5ef0fe1ce7db91ea7d3d1
Author: Daniil Osokin <daniil.osokin@itseez.com>
Date: Sun Aug 16 14:35:44 2015 +0300
Switched to factory
diff --git a/README.md b/README.md
index afadff2..bde857e 100644
--- a/README.md
+++ b/README.md
@@ -40,10 +40,10 @@ __Цель данной работы__ - реализовать набор пр
содержащий простейшую реализацию класса матриц. Предполагается, что он не
редактируется при реализации фильтров.
- Модуль `filters`(`./include/filters.hpp`, `./src/filters_opencv.cpp`,
- `./src/filters_fabrics.cpp`), содержащий объявление абстрактного класса
+ `./src/filters_factory.cpp`), содержащий объявление абстрактного класса
фильтров (`filters.hpp`) и его наследника, который реализует перечисленные
фильтры средствами библиотеки OpenCV (`filters_opencv.cpp`), а также метод
- создания конкретной реализации класса фильтров (`filters_fabrics.cpp`).
+ создания конкретной реализации класса фильтров (`filters_factory.cpp`).
- Тесты для класса матриц и фильтров (`matrix_test.cpp`, `filters_test.cpp`).
- Пример использования фильтра (`matrix_sample.cpp`).
@@ -489,11 +489,11 @@ __Примечание:__ генератор проекта должен сов
значение, соответствующее вашей реализации фильтра. Назовите его
согласно вашей фамилии `YOUR_NAME`. Указанное перечисление используется
при прогоне одних и тех же тестов на всех реализациях класса фильтров.

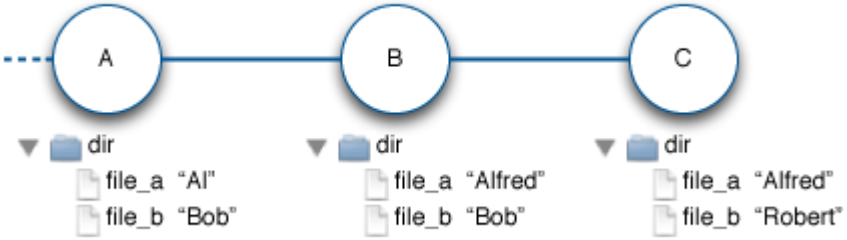
Терминология

Патчи

- Патч — это простой текстовый файл, его можно наложить при помощи инструментов (`patch`).
- Один патч может содержать изменения сразу нескольких файлов в разных директориях.
- Люди могут обмениваться изменениями, посылая друг другу патчи.
- Патч — это **атомарное изменение** проекта!

Патчи и СКВ

- СКВ — это своего рода БД патчей, ее называют **репозиторием.**
- Патчи, помещенные в СКВ называются **commit.**
- Последовательности **commit** называются **changeset.**



История изменений

ddc4a1d — **Readme bug fixes.**

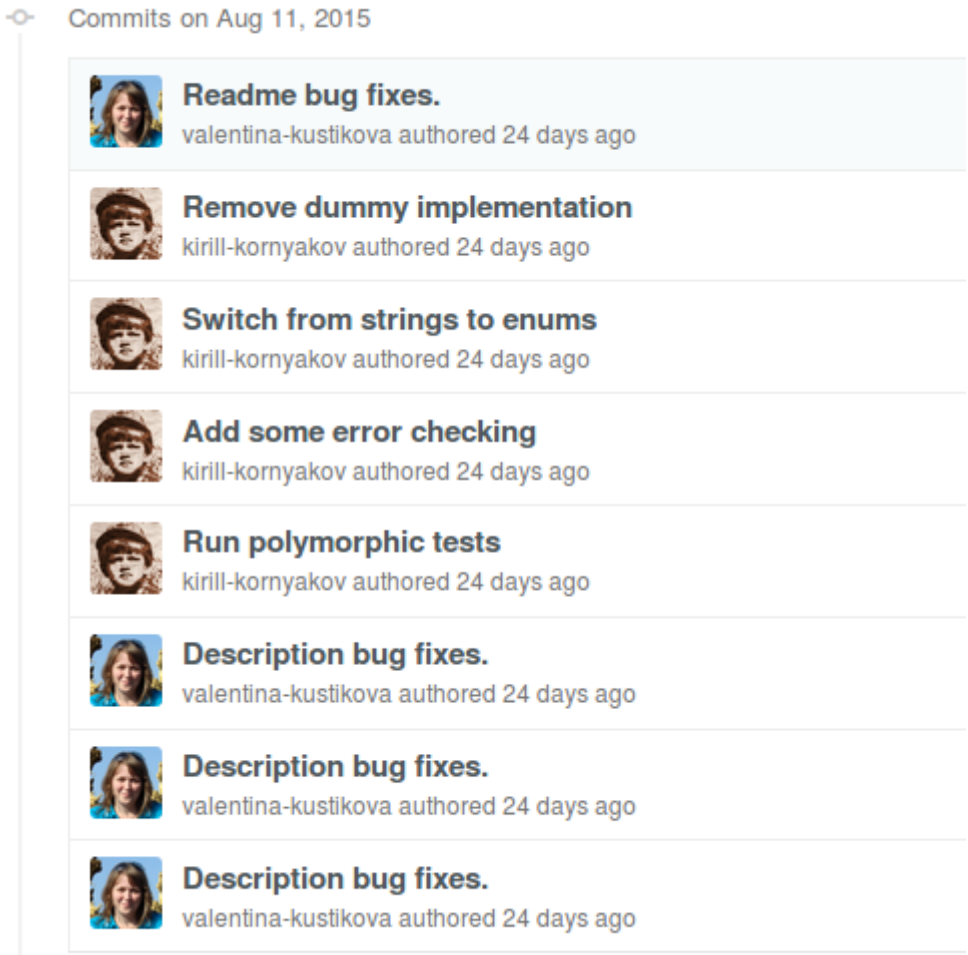
```
- README.md
```

f9e76e6 — **Remove dummy implementation**

```
- include/filters.hpp
- samples/matrix_sample.cpp
- src/filters_dummy.cpp
- src/filters_fabrics.cpp
- test/filters_test.cpp
```

aa1611b — **Switch from strings to enums**

```
- include/filters.hpp
```



Последовательность патчей — это

```
- samples/matrix_sample.cpp
- src/filters_fabrics.cpp
- test/filters_test.cpp
```

8e8b21b — Add some error checking

```
- include/filters.hpp
- src/filters_fabrics.cpp
- test/filters_test.cpp
```

Визуализация истории изменений

- http://www.youtube.com/watch?v=ToD91PYaQOU
- Сделано при помощи gource

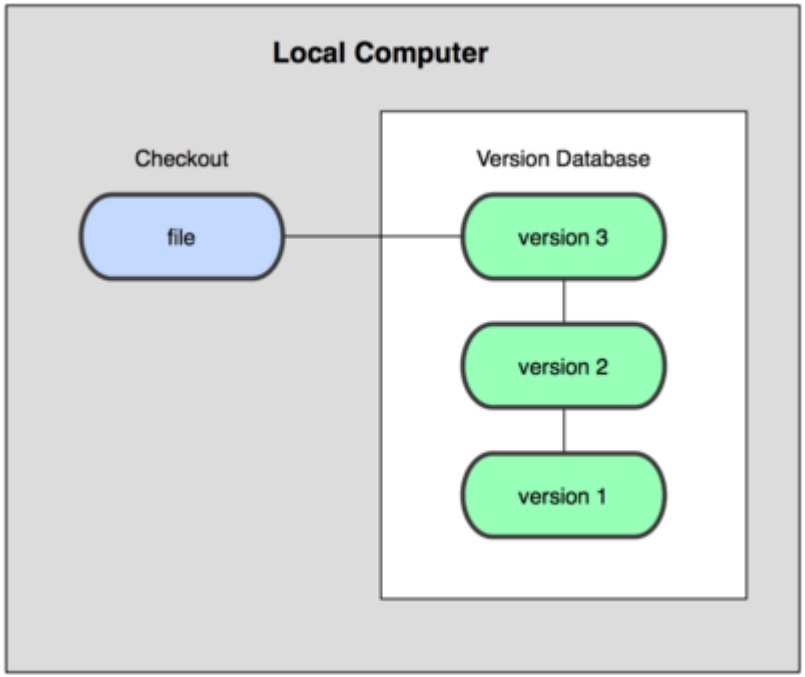
Три поколения СКВ

Generation	Networking	Operations	Concurrency	Examp
First	None	One file at a time	Locks	RCS,
Second	Centralized	Multi-file	Merge before commit	CVS, Subv Sour Tear Four Serv

Generation	Networking	Operations	Concurrency	Examp
Third	Distributed	Changesets	Commit before merge	Git, Merc Baza

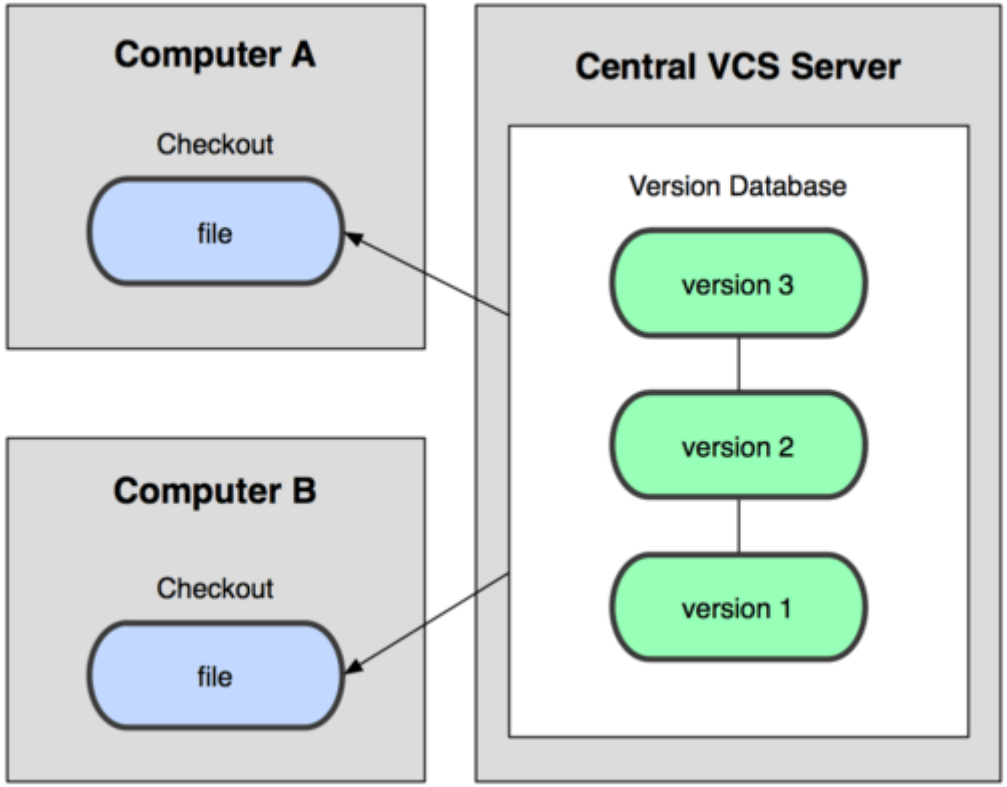
Eric Sink "A History of Version Control"

Три поколения СКВ: Локальные



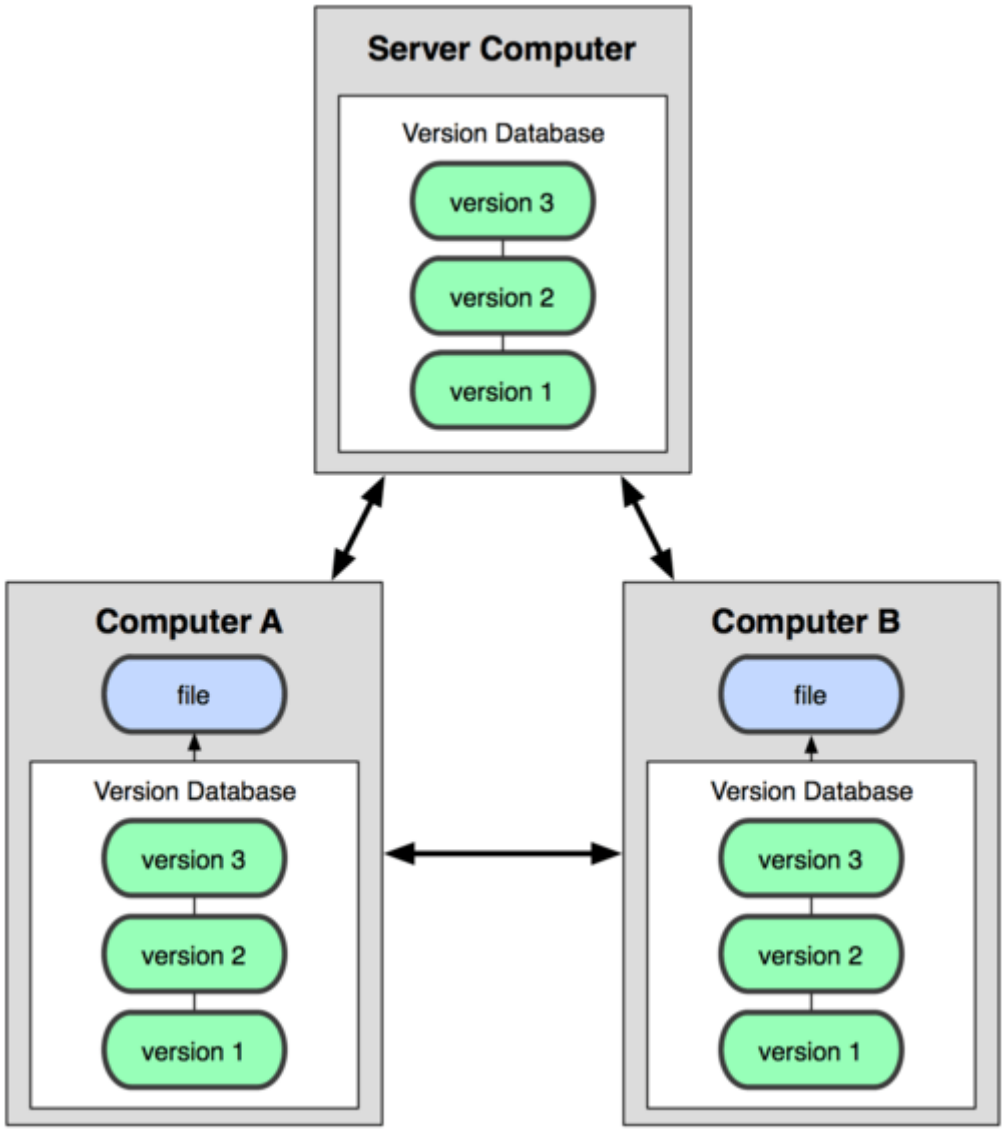
Примеры: RCS, SCCS

Три поколения СКВ: Централизованные



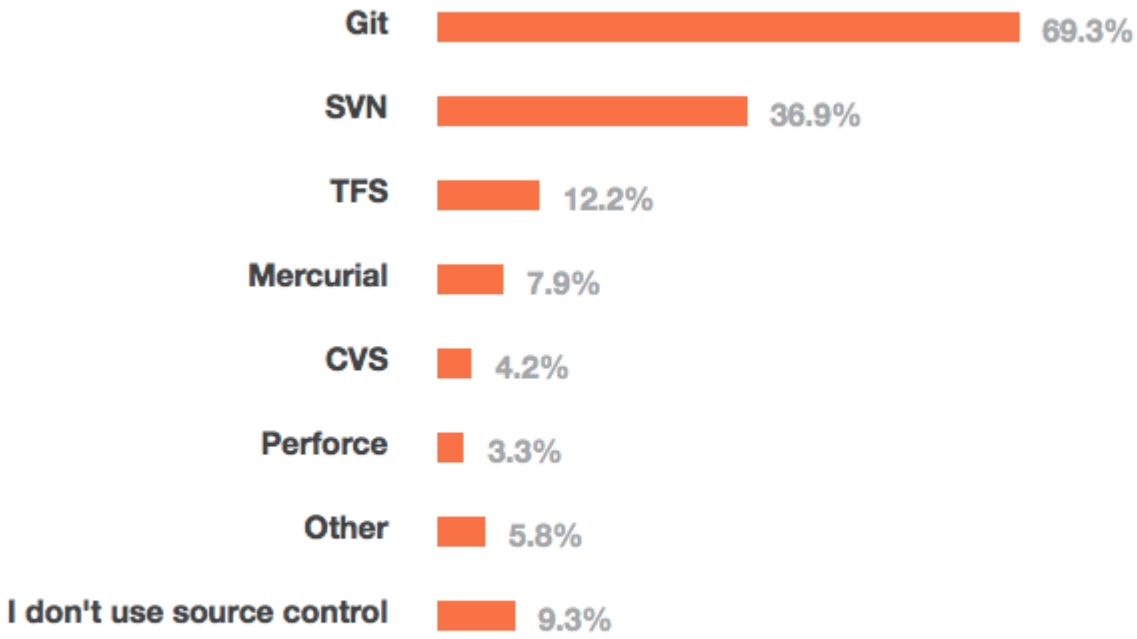
Примеры: Subversion, CVS

Три поколения СКВ: Распределенные


Server Computer
Version Database
version 3
version 2
version 1
Computer A
file
Version Database
version 3
version 2
version 1
Computer B
file
Version Database
version 3
version 2
version 1

- Примеры: Git, Mercurial
- Фактически стали стандартом де-факто
- Сильные стороны:
  - *Допускают локальную работу (коммиты без наличия интернет)*
  - *Упрощают слияние (а значит параллельную разработку)*
  - *Дают максимальную свободу по организации рабочего процесса (workflow)*

Популярные СКВ



16,694 responses

Stack Overflow Developer Survey 2015

- Использование в ИТ-проектах:
  - *Фундаментальный инструмент разработки*
  - *Также используется для: файлы конфигурации, документация, тестовые данные и пр.*

Git

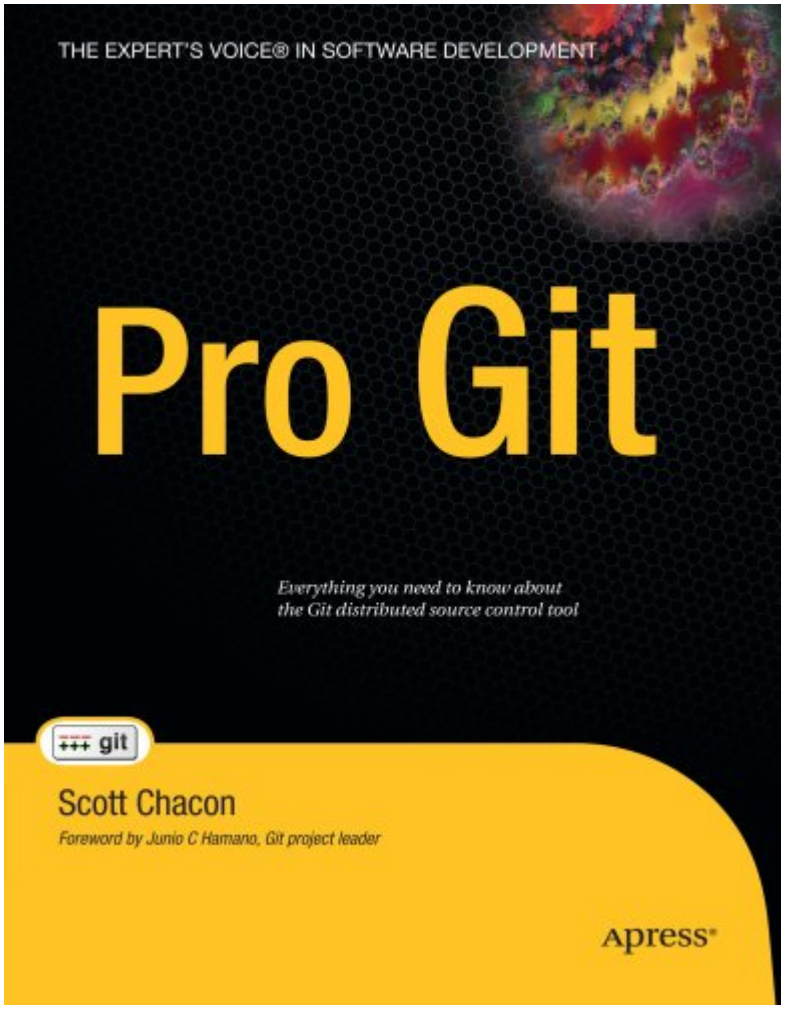
- Разработан Линусом Торвальдсом для работы над ядром Linux в 2005 году.
- В настоящее время поддерживается Джунио Хамано, сотрудником Google.

- Не очень прост в освоении, однако очень быстрый и функциональный.
- Имеет наиболее "сильное" сообщество, инструментальную поддержку.
- Огромное количество информации в интернет:

инструкции, уроки, статьи

- Официальный сайт проекта: http://www.git-scm.org.

Pro Git

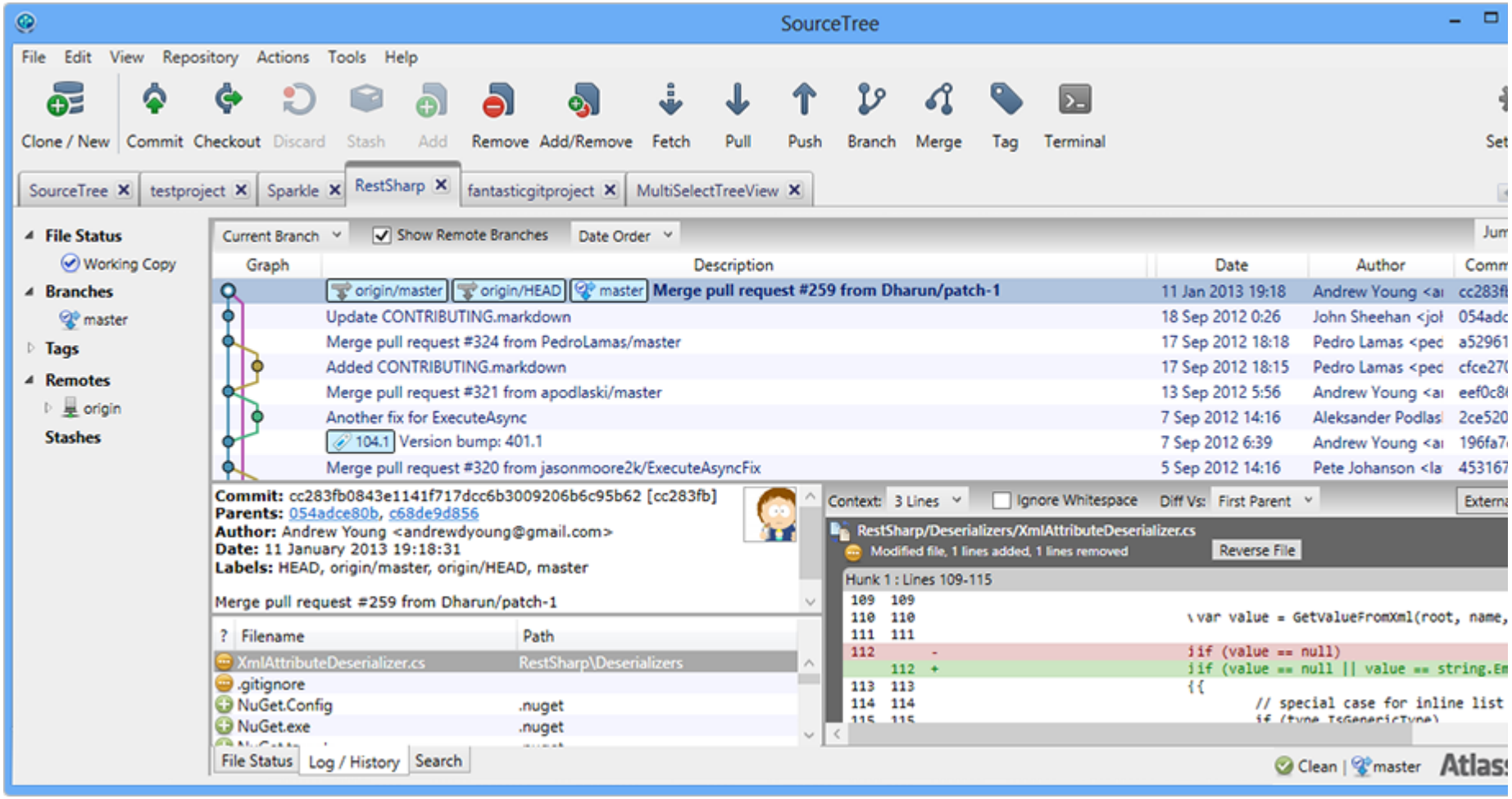


- Лучшая книга про Git
- Доступна бесплатно
- Переведена на русский язык
- Единственный способ по-настоящему понять Git — это узнать как он работает
- **Нужно прочесть хотя бы первые 100 страниц**

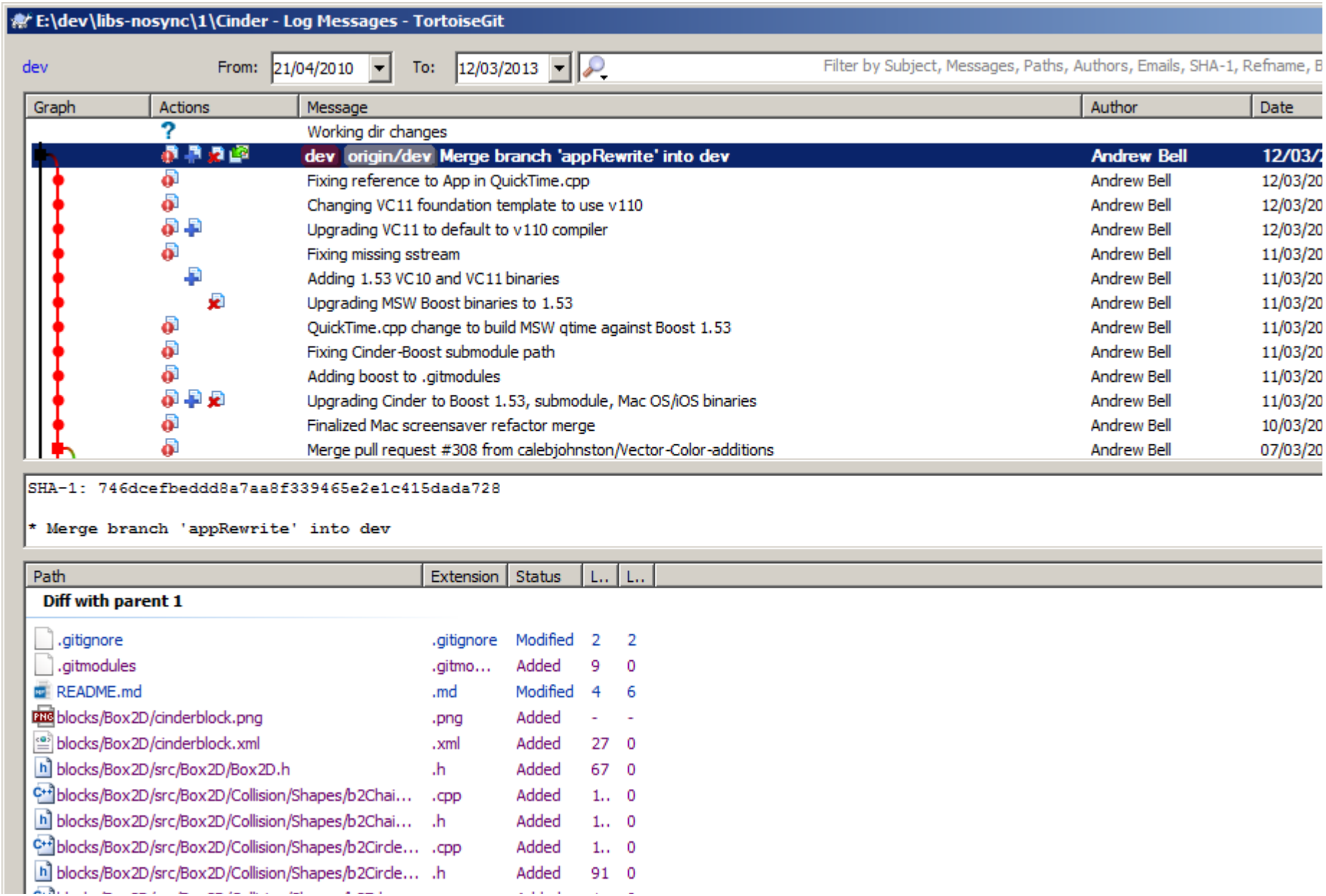
Как сказал Евклид египетскому царю Птолемею:

«Царской дороги в геометрии нет!»

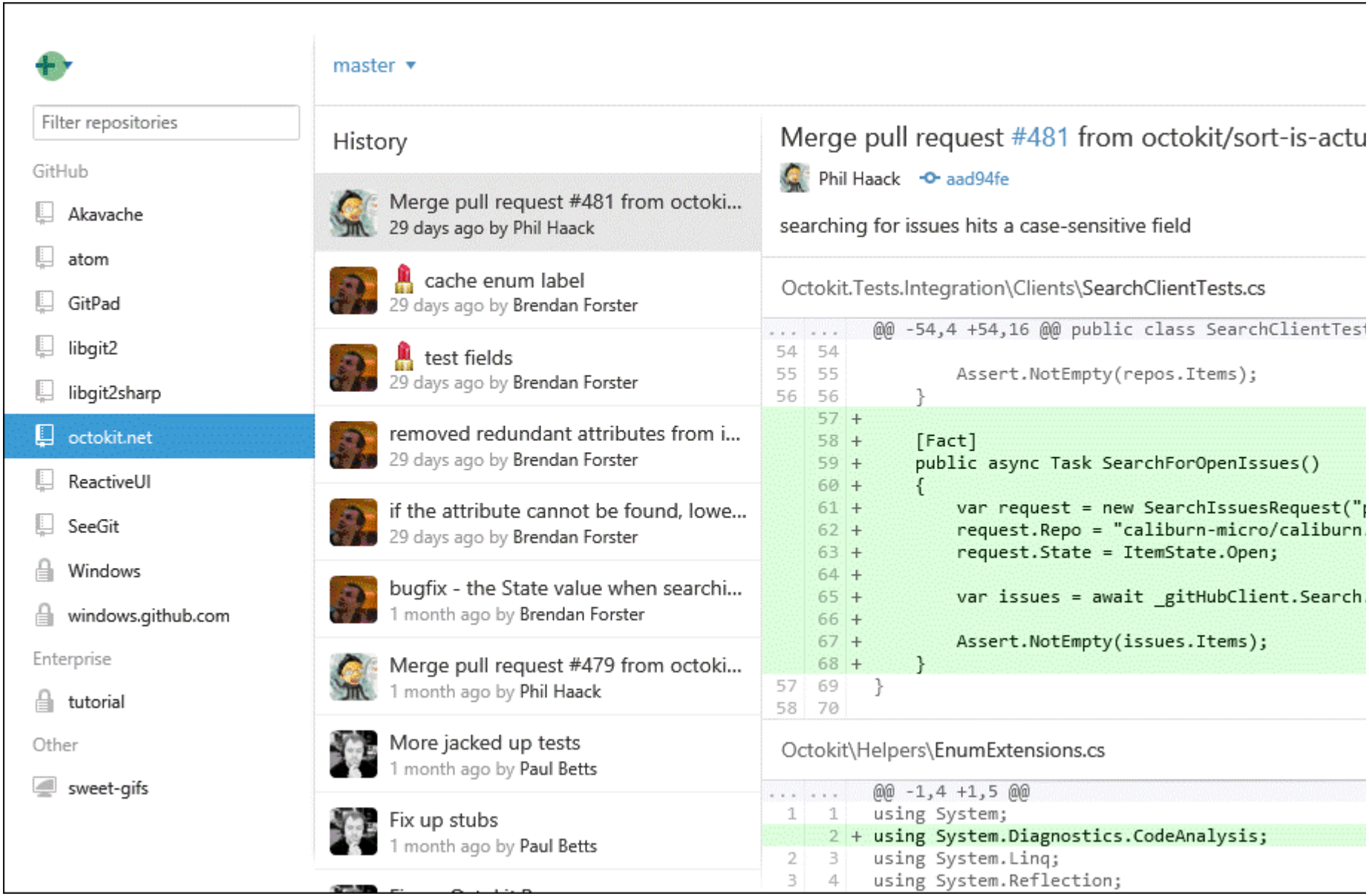
Atlassian SourceTree


SourceTree
File Edit View Repository Actions Tools Help
Clone / New
Commit
Checkout
Discard
Stash
Add
Remove
Add/Remove
Fetch
Pull
Push
Branch
Merge
Tag
Terminal
SourceTree
testproject
Sparkle
RestSharp
fantasticgitproject
MultiSelectTreeView
File Status
Working Copy
Branches
master
Tags
Remotes
origin
Stashes
Current Branch
Show Remote Branches
Date Order
Graph
Description
Date
Author
origin/master
origin/HEAD
master
Merge pull request #259 from Dharun/patch-1
11 Jan 2013 19:18
Andrew Young
Update CONTRIBUTING.markdown
18 Sep 2012 0:26
John Sheehan
Merge pull request #324 from PedroLamas/master
17 Sep 2012 18:18
Pedro Lamas
Added CONTRIBUTING.markdown
17 Sep 2012 18:15
Pedro Lamas
Merge pull request #321 from apodlaski/master
13 Sep 2012 5:56
Andrew Young
Another fix for ExecuteAsync
7 Sep 2012 14:16
Aleksander Podlas
104.1
Version bump: 401.1
7 Sep 2012 6:39
Andrew Young
Merge pull request #320 from jasonmoore2k/ExecuteAsyncFix
5 Sep 2012 14:16
Pete Johanson
Commit: cc283fb0843e1141f717dcc6b3009206b6c95b62 [cc283fb]
Parents: 054adce80b, c68de9d856
Author: Andrew Young <andrewdyoung@gmail.com>
Date: 11 January 2013 19:18:31
Labels: HEAD, origin/master, origin/HEAD, master
Merge pull request #259 from Dharun/patch-1
Filename
Path
XmlAttributeDeserializer.cs
RestSharp\Deserializers
.gitignore
NuGet.Config
.nuget
NuGet.exe
.nuget
Context: 3 Lines
Ignore Whitespace
Diff Vs: First Parent
RestSharp/Deserializers/XmlAttributeDeserializer.cs
Modified file, 1 lines added, 1 lines removed
Reverse File
Hunk 1 : Lines 109-115
109 109
110 110 var value = GetValueFromXml(root, name,
111 111
112 - if (value == null)
112 + if (value == null || value == string.Em
113 113 {
114 114 // special case for inline list
File Status
Log / History
Search
Clean
master
Atlas

Tortoise Git


E:\dev\libs-nosync\1\Cinder - Log Messages - TortoiseGit
dev
From: 21/04/2010
To: 12/03/2013
Filter by Subject, Messages, Paths, Authors, Emails, SHA-1, Refname, B
Graph
Actions
Message
Author
Date
Working dir changes
dev origin/dev Merge branch 'appRewrite' into dev
Andrew Bell
12/03/
Fixing reference to App in QuickTime.cpp
Andrew Bell
12/03/20
Changing VC11 foundation template to use v110
Andrew Bell
12/03/20
Upgrading VC11 to default to v110 compiler
Andrew Bell
12/03/20
Fixing missing sstream
Andrew Bell
11/03/20
Adding 1.53 VC10 and VC11 binaries
Andrew Bell
11/03/20
Upgrading MSW Boost binaries to 1.53
Andrew Bell
11/03/20
QuickTime.cpp change to build MSW qtime against Boost 1.53
Andrew Bell
11/03/20
Fixing Cinder-Boost submodule path
Andrew Bell
11/03/20
Adding boost to .gitmodules
Andrew Bell
11/03/20
Upgrading Cinder to Boost 1.53, submodule, Mac OS/iOS binaries
Andrew Bell
11/03/20
Finalized Mac screensaver refactor merge
Andrew Bell
10/03/20
Merge pull request #308 from calebjohnston/Vector-Color-additions
Andrew Bell
07/03/20
SHA-1: 746dcefbeddd8a7aa8f339465e2e1c415dada728
* Merge branch 'appRewrite' into dev
Path
Extension
Status
L..
L..
Diff with parent 1
.gitignore .gitignore Modified 2 2
.gitmodules .gitmo... Added 9 0
README.md .md Modified 4 6
blocks/Box2D/cinderblock.png .png Added - -
blocks/Box2D/cinderblock.xml .xml Added 27 0
blocks/Box2D/src/Box2D/Box2D.h .h Added 67 0
blocks/Box2D/src/Box2D/Collision/Shapes/b2Chai... .cpp Added 1.. 0
blocks/Box2D/src/Box2D/Collision/Shapes/b2Chai... .h Added 1.. 0
blocks/Box2D/src/Box2D/Collision/Shapes/b2Circle... .cpp Added 1.. 0
blocks/Box2D/src/Box2D/Collision/Shapes/b2Circle... .h Added 91 0

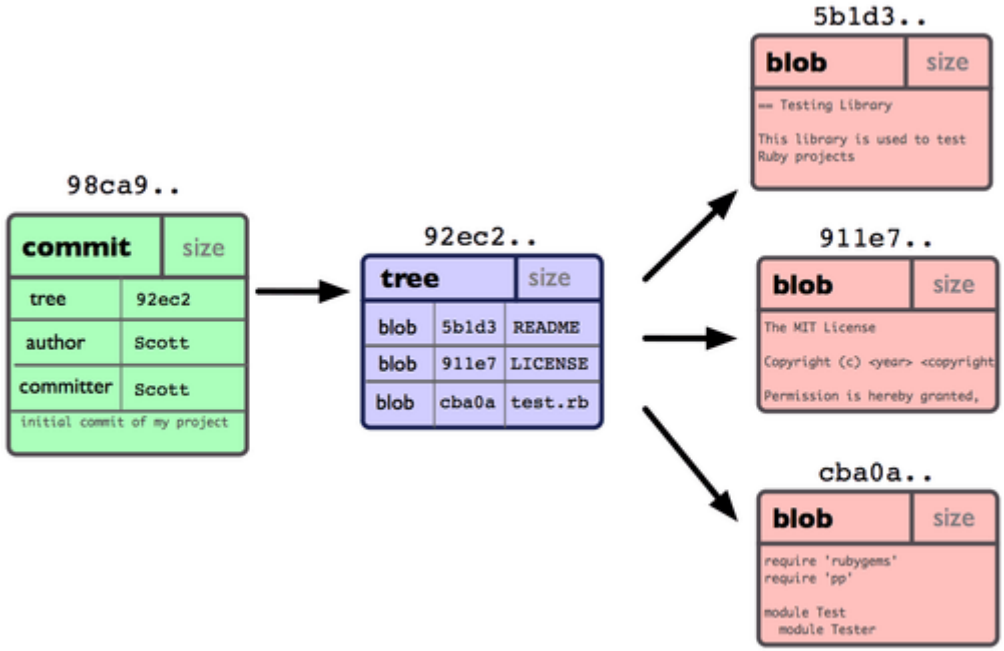
GitHub Desktop


master
Filter repositories
GitHub
Akavache
atom
GitPad
libgit2
libgit2sharp
octokit.net
ReactiveUI
SeeGit
Windows
windows.github.com
Enterprise
tutorial
Other
sweet-gifs
History
Merge pull request #481 from octoki...
29 days ago by Phil Haack
cache enum label
29 days ago by Brendan Forster
test fields
29 days ago by Brendan Forster
removed redundant attributes from i...
29 days ago by Brendan Forster
if the attribute cannot be found, lowe...
29 days ago by Brendan Forster
bugfix - the State value when searchi...
1 month ago by Brendan Forster
Merge pull request #479 from octoki...
1 month ago by Phil Haack
More jacked up tests
1 month ago by Paul Betts
Fix up stubs
1 month ago by Paul Betts
Merge pull request #481 from octokit/sort-is-actu
Phil Haack
aad94fe
searching for issues hits a case-sensitive field
Octokit.Tests.Integration\Clients\SearchClientTests.cs
@@ -54,4 +54,16 @@ public class SearchClientTest
Assert.NotEmpty(repos.Items);
}
[Fact]
public async Task SearchForOpenIssues()
{
var request = new SearchIssuesRequest("
request.Repo = "caliburn-micro/caliburn
request.State = ItemState.Open;
var issues = await _gitHubClient.Search
Assert.NotEmpty(issues.Items);
}
}
Octokit\Helpers\EnumExtensions.cs
@@ -1,4 +1,5 @@
using System;
using System.Diagnostics.CodeAnalysis;
using System.Linq;
using System.Reflection;

Command Line Interface!

```
~> git help
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p|--paginate|--no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

The most commonly used git commands are:
   add        Add file contents to the index
   bisect     Find by binary search the change that introduced a bug
   branch     List, create, or delete branches
   checkout   Checkout a branch or paths to the working tree
   clone      Clone a repository into a new directory
   commit     Record changes to the repository
   diff       Show changes between commits, commit and working tree, etc
   fetch      Download objects and refs from another repository
   grep       Print lines matching a pattern
   init       Create an empty Git repository or reinitialize an existing one
   log        Show commit logs
   merge      Join two or more development histories together
   mv         Move or rename a file, a directory, or a symlink
   pull       Fetch from and integrate with another repository or a local branch
   push       Update remote refs along with associated objects
   rebase     Forward-port local commits to the updated upstream head
   reset      Reset current HEAD to the specified state
   rm         Remove files from the working tree and from the index
   show       Show various types of objects
   status     Show the working tree status
```

Git objects



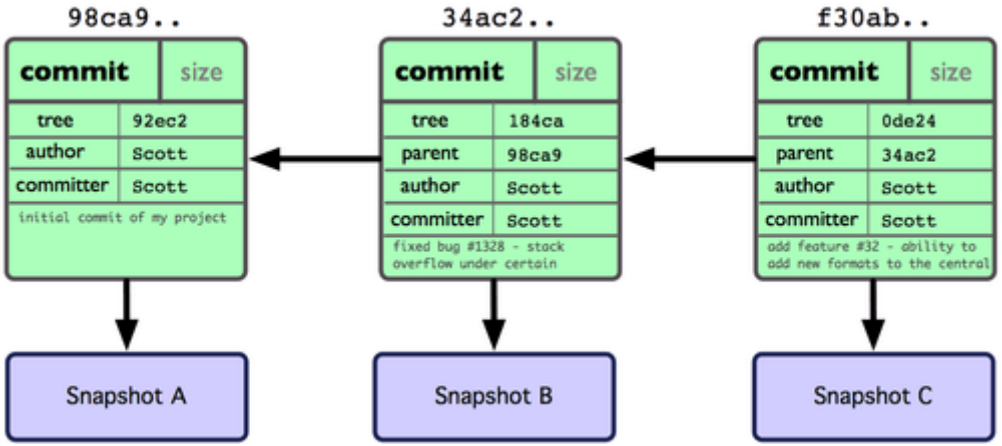
- По сути это *внутреннее* представление патча
- Пользователю приходится работать только с коммитами

(слава богу!)

Показать содержимое коммита:

```
$ git show --raw dc2ca9d95c

commit dc2ca9d95cbd5586e9e5ef0fe1ce7db91ea7d3d1
Author: Daniil Osokin <daniil.osokin@itseez.com>
Date:   Sun Aug 16 14:35:44 2015 +0300

    Switched to factory

:100644 100644 afadff2... bde857e... M  README.md
:100644 000000 c977bf3... 0000000... D  src/filters_fabrics.cpp
:000000 100644 0000000... c977bf3... A  src/filters_factory.cpp
```

Git commits



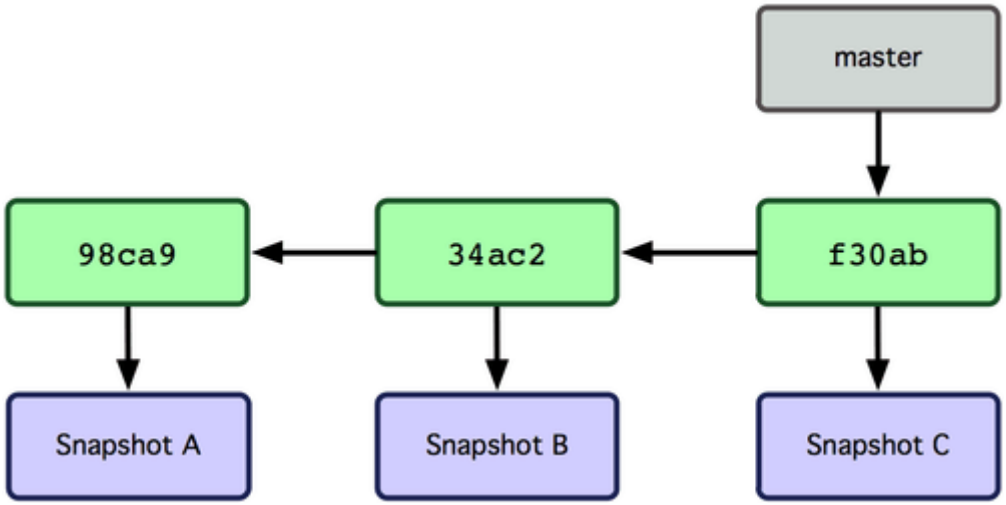
Вывести историю изменений:

```
$ git log

commit aaa321be9191da60ad52c2bc41bd749ed546b409
Merge: 98fce98 3c1d15a
Author: Valentina <valentina-kustikova@users.noreply.github.com>
Date:   Thu Aug 13 10:14:47 2015 +0300
```

```
    Merge pull request #11 from valentina-kustikova/master

    Practice description (bug fixes).

commit 3c1d15a1bf366864593f2320fa9a0e6cf3586f52
Author: valentina-kustikova <valentina.kustikova@gmail.com>
Date:   Thu Aug 13 10:08:59 2015 +0300

    Practice description (bug fixes).
```

Понятие ветки (branch)

- Ветка в Git'е — это просто **указатель** на один из коммитов.
- Есть соглашение, что имя **`master`** используется для ветки,
  указывающей на последнее актуальное состояние проекта.



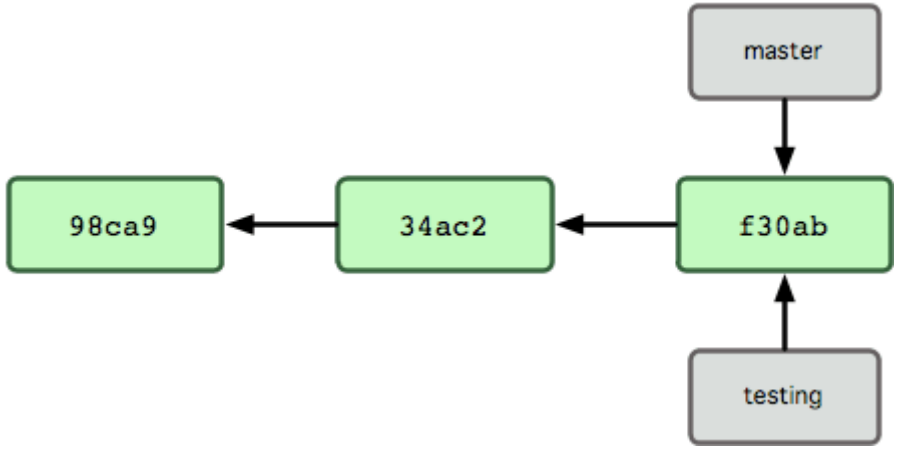
Вывести список существующих веток:

```
$ git branch
* master
```

Git `branch`

Создать новую ветку с именем `testing` (указатель на коммит!):

```
$ git branch testing
```

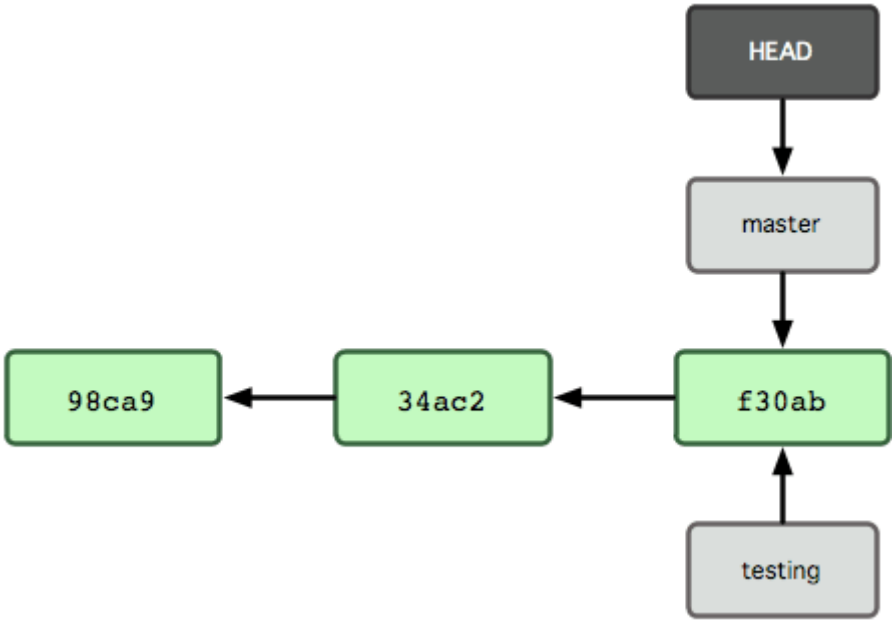


Текущий список веток:

```
$ git branch
* master
  testing
```

HEAD

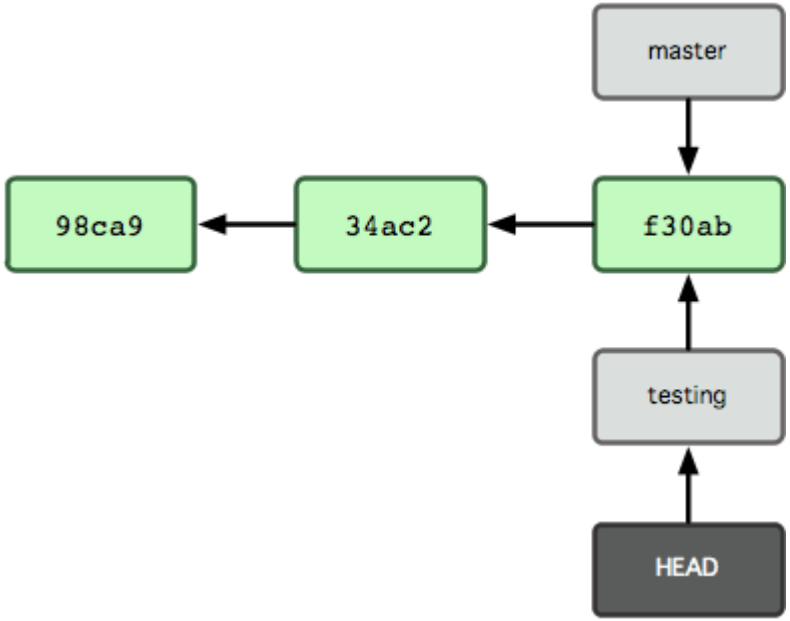
- HEAD — специальный указатель, ссылающийся на локальную ветку, на которой вы находитесь.
- Это просто алиас для текущей ветки, введенный для удобства.


HEAD
master
98ca9
34ac2
f30ab
testing

Git checkout

Извлечь состояние репозитория, соответствующее ветке `testing`:

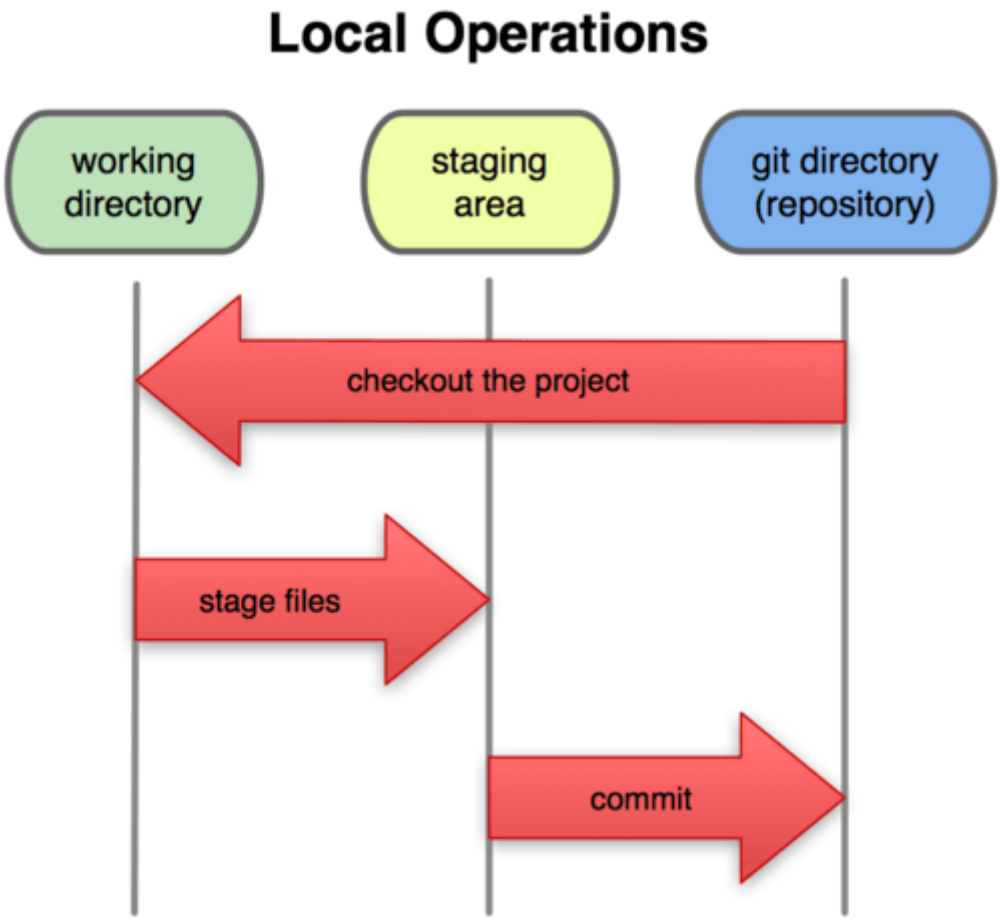
```
$ git checkout testing
```



Вывести список существующих веток:

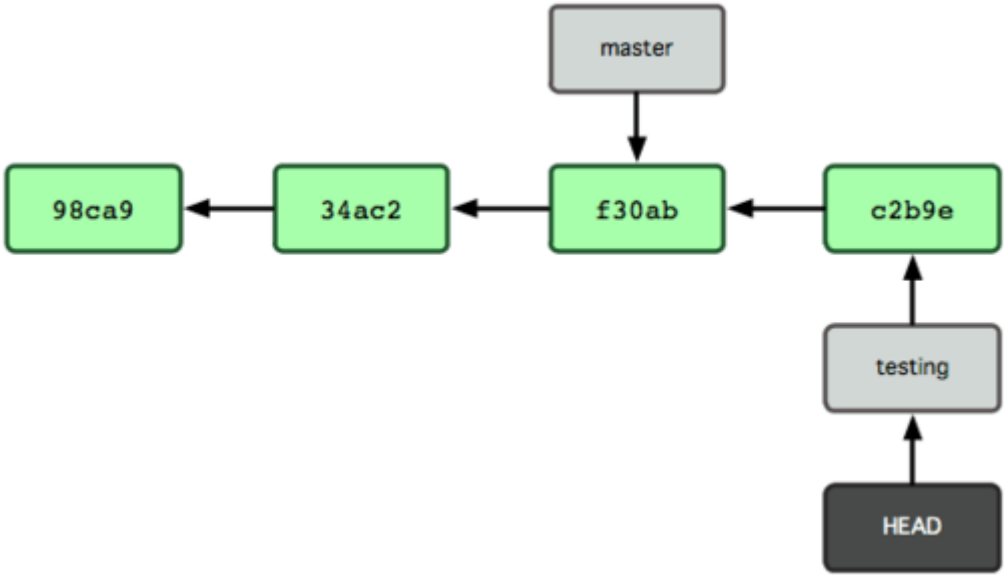
```
$ git branch
  master
* testing
```

Три состояния файлов



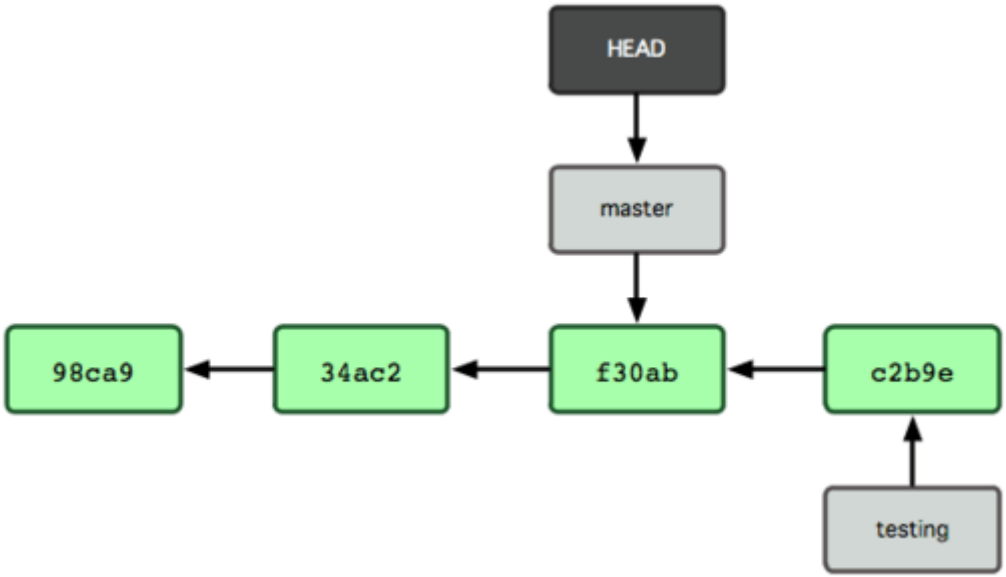
Git commit

```
$ vim README.md
$ git add README.md
$ git commit -m 'Made a change'
```



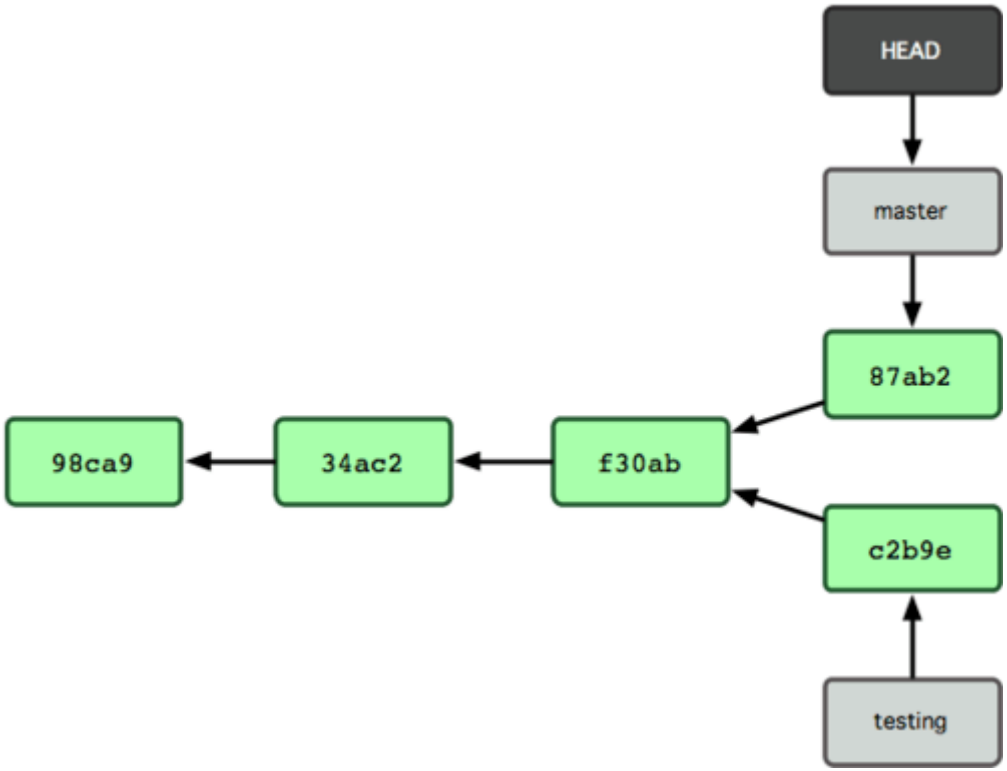
Go back to `master`

```
$ git checkout master
```

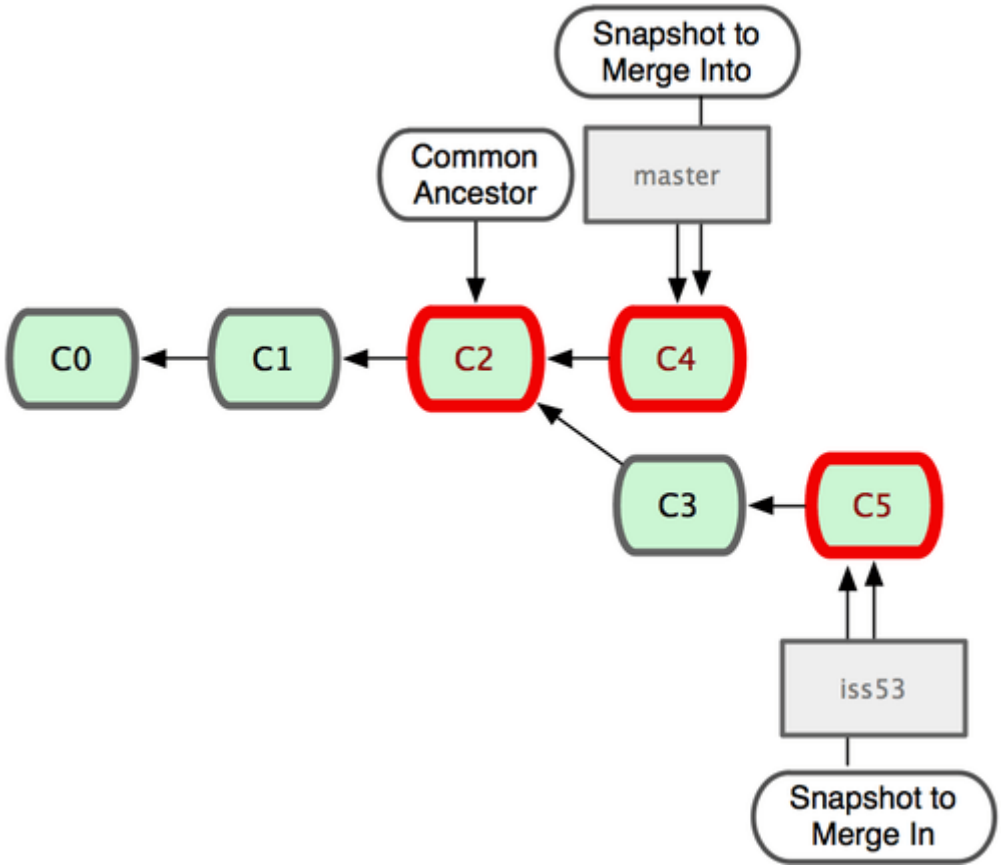


Make a commit to `master`

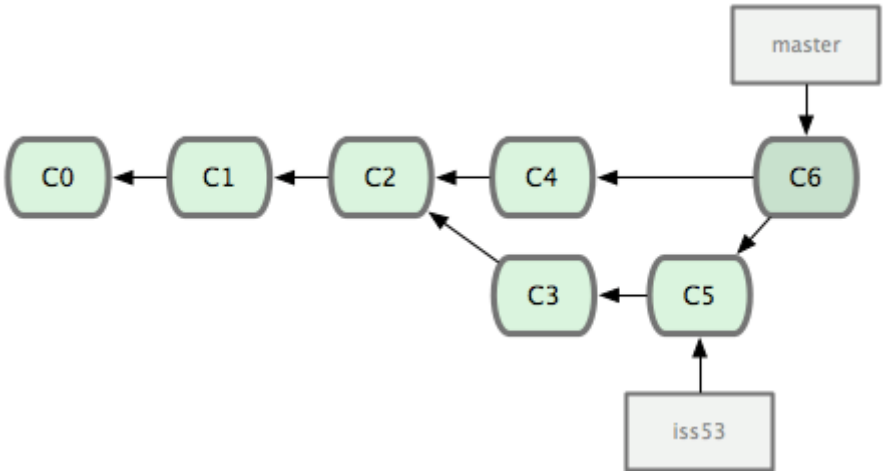
```
$ vim main.cpp

$ git add main.cpp
$ git commit -m 'Made other changes'

# Или можно сделать так
$ git status
$ git commit -a -m 'Made other changes'
```


HEAD
master
87ab2
98ca9
34ac2
f30ab
c2b9e
testing

Merging


Snapshot to Merge Into
Common Ancestor
master
C0
C1
C2
C4
C3
C5
iss53
Snapshot to Merge In

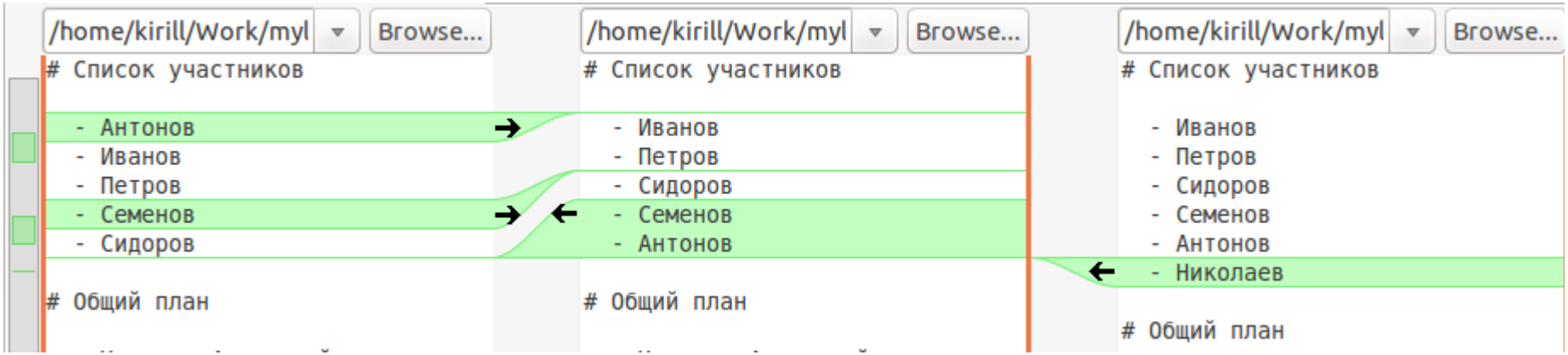
Merging



- C6 — это так называемый *merge commit*
- Он основан не на каком-то патче, он указывает на состояние проекта,
  в котором наложены патчи обоих ветвей (`master` и

testing).

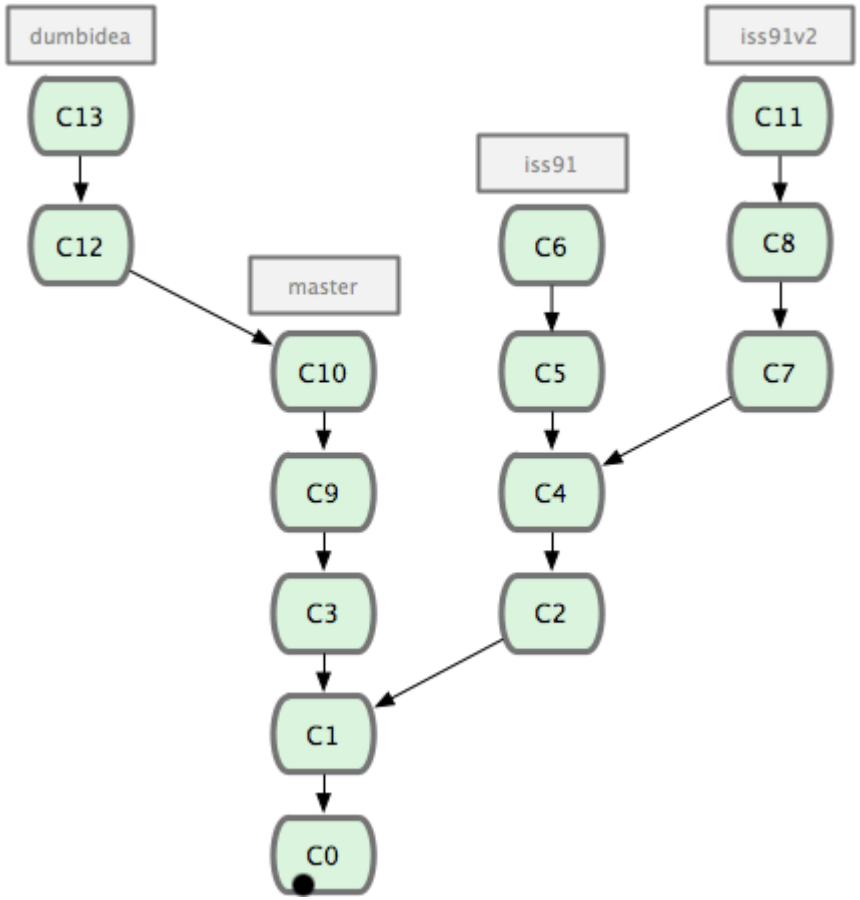
Merge Conflicts



- Возникают когда несколько участников отредактировали одинаковые строки,
  или когда это произошло в разных ветках.
- Разрешаются человеком при помощи инструментов (`git mergetool`).

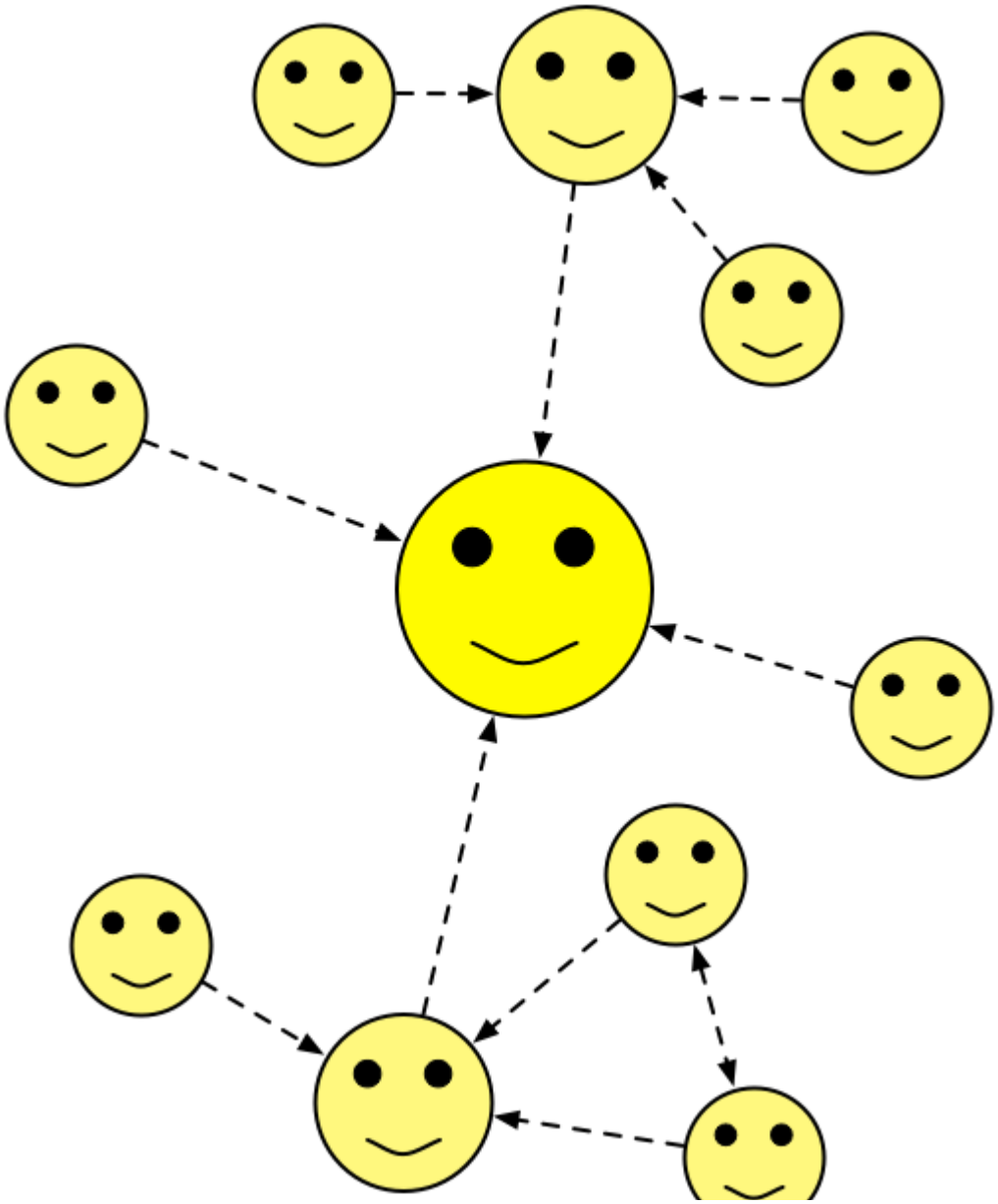
- В реальности довольно редкая ситуация, если соблюдать практики:
  - *Грамотное распределение задач*
  - *Частые коммиты, много маленьких веток, частая интеграция*

Multiple branches

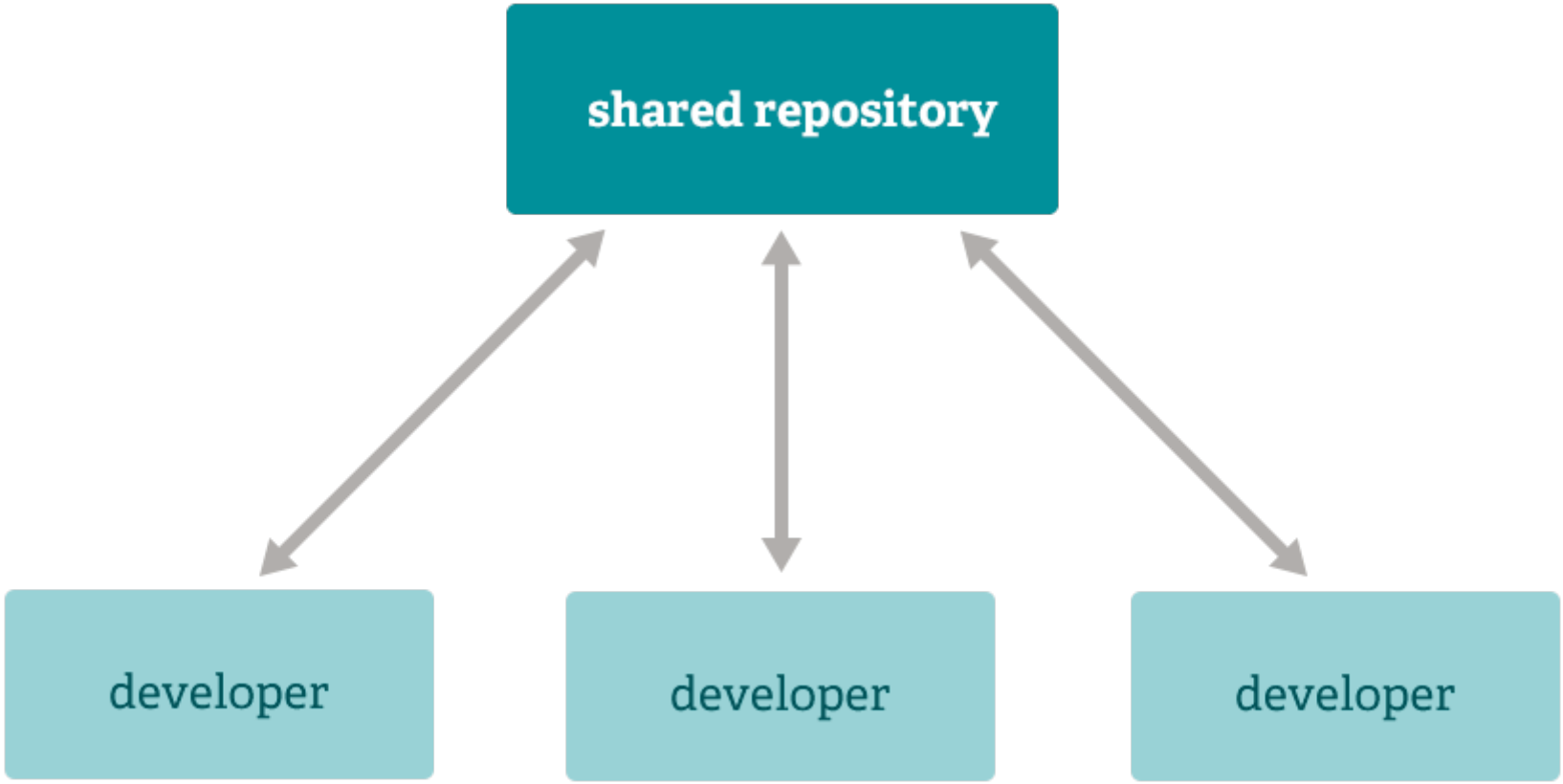


- Даже у одного разработчика может быть несколько активных веток.
- Правильно создавать отдельную ветку на каждую логически независимую задачу.
- Долгоживущие ветки — это неправильно, они быстро устаревают.

Распределенная работа

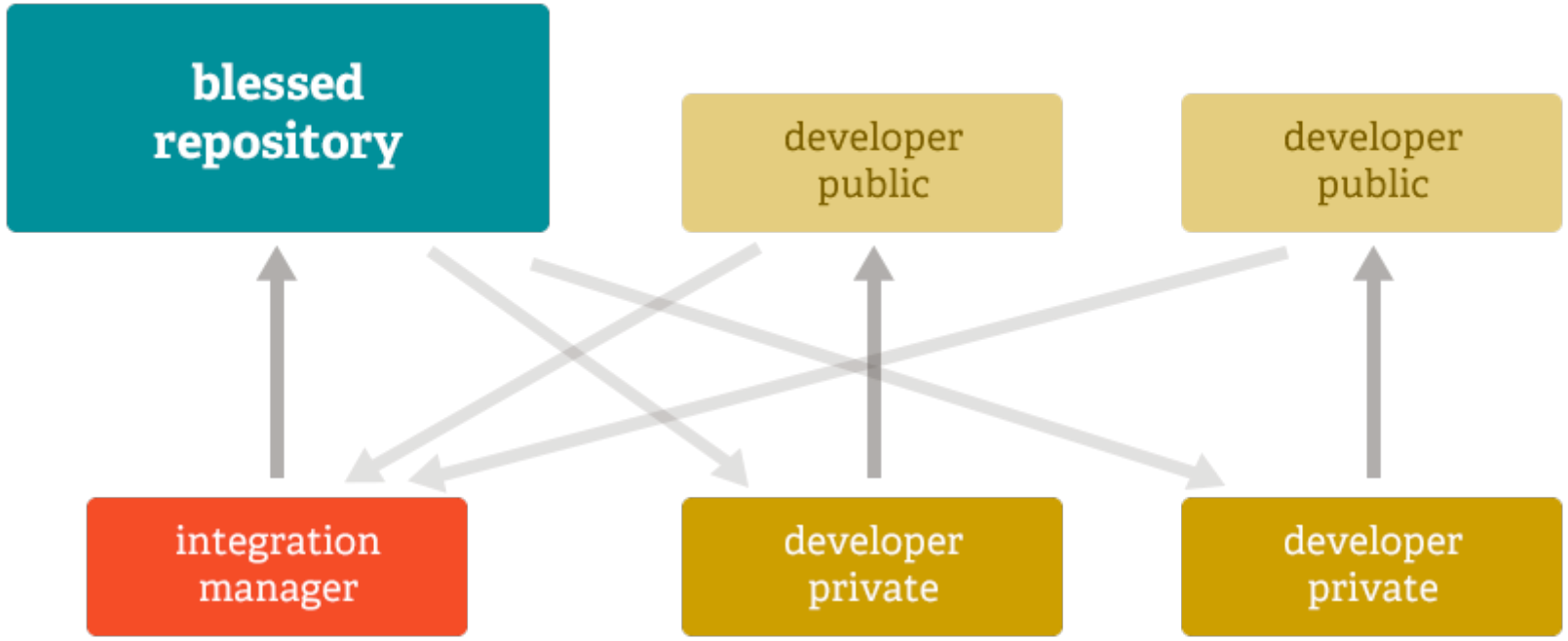


Centralized Workflow



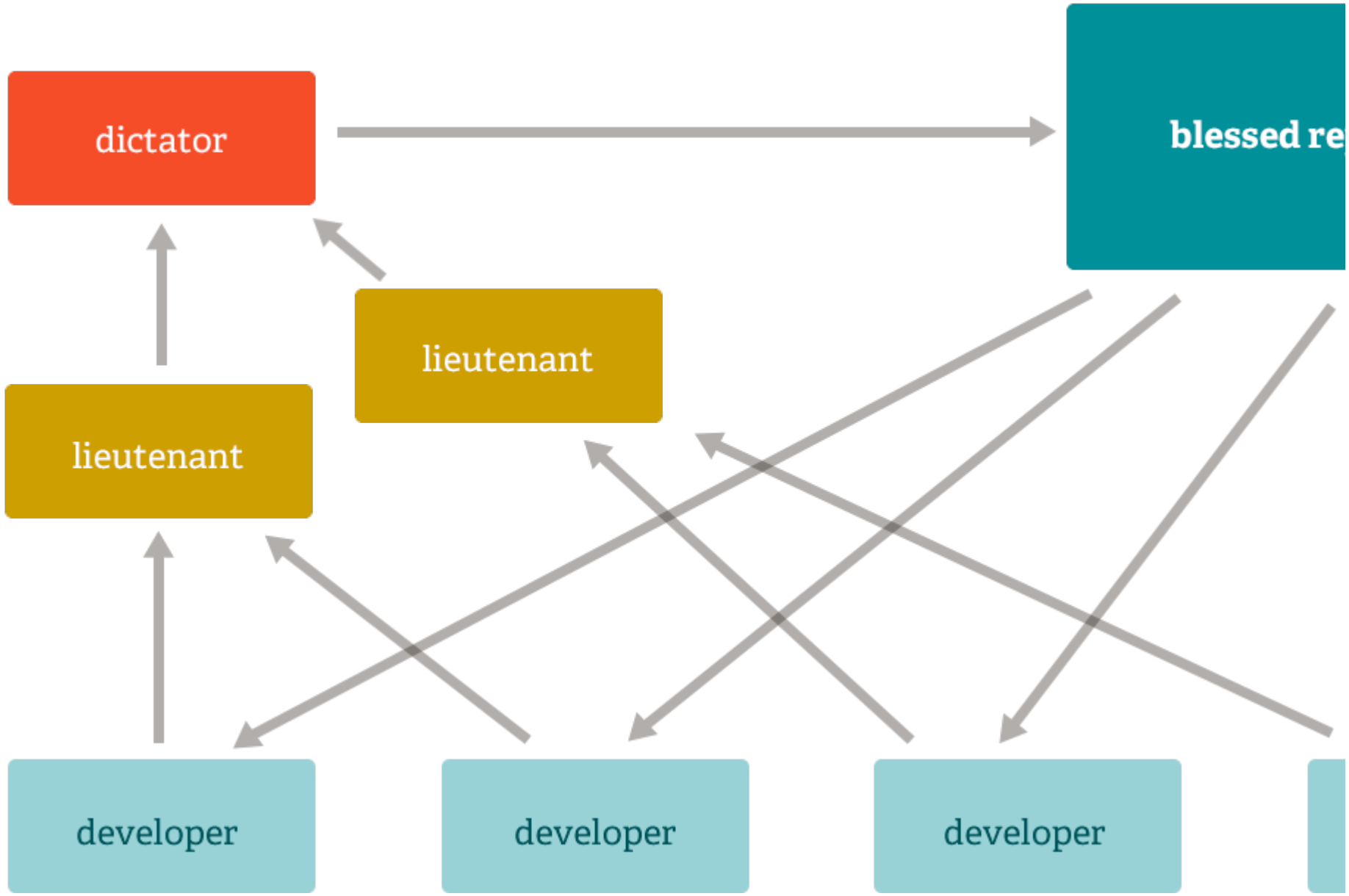
Плюсы и минусы данного подхода?

Integration Manager Workflow

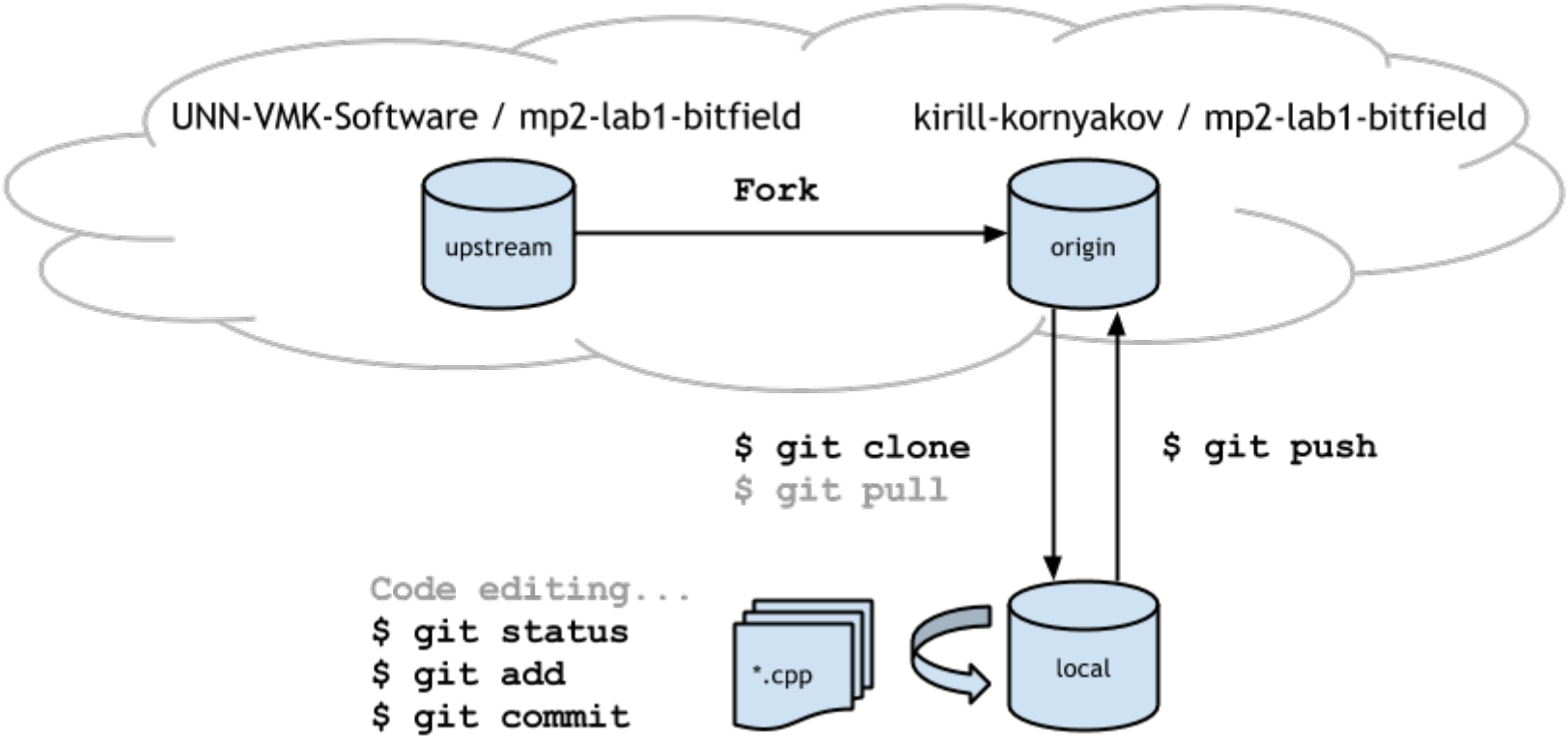


Плюсы и минусы данного подхода?

Dictator and Lieutenants Workflow


dictator
blessed re
lieutenant
lieutenant
developer
developer
developer

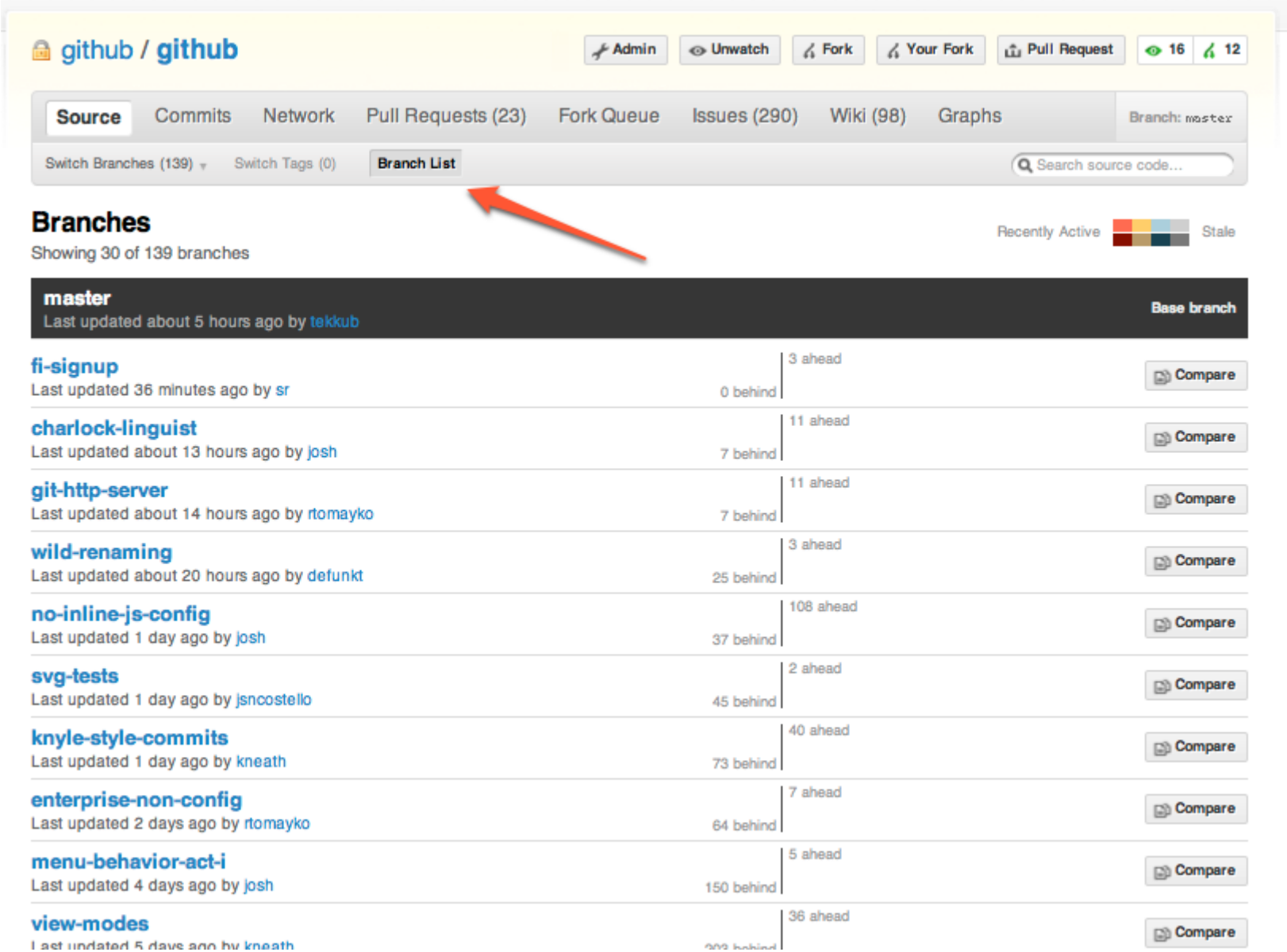
Triangular Workflow (GitHub)




```
$ cd mp2-lab1-bitfield
```

```
$ git remote -v
origin  https://github.com/kirill-kornyakov/mp2-lab1-bitfield.git (fetch)
origin  https://github.com/kirill-kornyakov/mp2-lab1-bitfield.git (push)
upstream  https://github.com/UNN-VMK-Software/mp2-lab1-bitfield.git (fetch)
upstream  https://github.com/UNN-VMK-Software/mp2-lab1-bitfield.git (push)
```

GitHub Flow


github / github
Admin
Unwatch
Fork
Your Fork
Pull Request
16
12
Source
Commits
Network
Pull Requests (23)
Fork Queue
Issues (290)
Wiki (98)
Graphs
Branch: master
Switch Branches (139)
Switch Tags (0)
Branch List
Search source code...
Branches
Showing 30 of 139 branches
Recently Active
Stale
master
Last updated about 5 hours ago by tekkub
Base branch
fi-signup
Last updated 36 minutes ago by sr
3 ahead
0 behind
Compare
charlock-linguist
Last updated about 13 hours ago by josh
11 ahead
7 behind
Compare
git-http-server
Last updated about 14 hours ago by rtomayko
11 ahead
7 behind
Compare
wild-renaming
Last updated about 20 hours ago by defunkt
3 ahead
25 behind
Compare
no-inline-js-config
Last updated 1 day ago by josh
108 ahead
37 behind
Compare
svg-tests
Last updated 1 day ago by jsncostello
2 ahead
45 behind
Compare
knyle-style-commits
Last updated 1 day ago by kneath
40 ahead
73 behind
Compare
enterprise-non-config
Last updated 2 days ago by rtomayko
7 ahead
64 behind
Compare
menu-behavior-act-i
Last updated 4 days ago by josh
5 ahead
150 behind
Compare
view-modes
36 ahead
Compare

GitHub Flow

GitHub Flow

Anything in the `master` branch is deployable.

1. Create branch
    - *To work on something new, create a descriptively named branch off of `master` (ie: `new-oauth2-scopes`).*
2. Develop in branch
    - *Commit to that branch locally and regularly push your work to the same named branch on the server.*
3. Open a pull request (ask for review)
    - *When you need feedback or help, or you think the*

branch is ready for merging, open a pull request.

4. Merge after review

    - *After someone else has reviewed and signed off on the feature, you can merge it into `master`.*

5. Deploy

    - *Once it is merged and pushed to `master`, you can and should deploy immediately.*

GitHub Flow

```
# Check that origin and upstream repositories are correctly defined
$ git remote -v

# Get the latest sources from the upstream repository
$ git remote update

# Checkout a new topic branch for development
$ git checkout -b adding-new-feature upstream/master

#
# Do some development...
#

# Check your changes
$ git status

# Commit your changes
$ git commit -a -m "Added a new feature"

# Push your changes to the origin
$ git push origin HEAD
```

VCS: Резюме

1. Системы контроля версий — центральный инструмент разработки
    - *Навигация по истории изменений*
    - *Централизованный доступ*
2. Распределенные СКВ фактически стали стандартом. Их сильные стороны:
    - *Допускают локальные коммиты (без наличия интернет или доступа к серверу)*
    - *Упрощают слияние (а значит параллельную разработку)*

    - *Дают максимальную свободу по организации рабочего процесса (workflow)*

3. Git не самая простая в освоении СКВ, однако очень функциональная,
   к тому же дает максимальную свободу по организации процесса разработки.

googletest

Google C++ Testing Framework

Вся правда о ручном тестировании

- Ключевые термины
  - ***Тест*** *— проверка, осуществляемая "руками"*
  - ***Тест-план*** *— документ со списком проверок*
  - *Отдел тестирования*
- **Профессия ручного тестировщика умирает!**
  - *Программисты несут ответственность за качество (пишут тесты!)*
  - *Google: Software Engineer in Test*

- Ручное тестирование все еще используется для:
  - *Тестирования GUI и UX (удобства использования)*
  - *Бета-тестирование с реальными пользователями*

Автоматические тесты

- **Тест** — это "обычная" функция, реализующая некоторый сценарий использования программных сущностей.

```
#include <gtest/gtest.h>

TEST(TBitField, can_set_bit)
{
    TBitField bf(10);
    EXPECT_EQ(0, bf.GetBit(3));

    bf.SetBit(3);
    EXPECT_NE(0, bf.GetBit(3));
}
```

- **Тестовая сборка (test suite)** — приложение с тестами (обычно консольное).

Тест пишется один раз, а запускается десятки тысяч раз!

Пример тестов на Java с использованием JUnit

```java
@Test
public void canAddNumbers()
{
    // Arrange
    ComplexNumber z1 = new ComplexNumber(1, 2);
    ComplexNumber z2 = new ComplexNumber(3, 4);

    // Act
    ComplexNumber sum = z1.add(z2);

    // Assert
    assertEquals(new ComplexNumber(4, 6), sum);
}

@Test
public void canMultiplyNumbers()
{
    // Arrange
```

```
    ComplexNumber z1 = new ComplexNumber(1, 2);
    ComplexNumber z2 = new ComplexNumber(3, 4);

    // Act
    ComplexNumber mult = z1.multiply(z2);

    // Assert
    assertEquals(new ComplexNumber(-5, 10), mult);
}
```

Фреймворки для Unit-тестирования

Значительно упрощают создание и запуск unit-тестов, позволяют придерживаться единого стиля.

1. xUnit — общее обозначение для подобных фреймворков.
2. Бесплатно доступны для большинства языков:
    - *C/C++*: *CUnit*, *CPPUnit*, *GoogleTest*
    - *Java*: *JUnit*
    - *.NET*: *NUnit*

3. **Встроены в современные языки:**

- *D*, *Python*, *Go*

Типичные возможности

1. Удобное добавление тестов
    - *Простая регистрация новых тестов*
    - *Набор функций-проверок (`assert`)*
    - *Общие инициализации и деинициализации*
2. Удобный запуск тестов
    - *Пакетный режим*
    - *Возможность фильтрации тестов по именам*
3. Часто допускают интеграцию с IDE
4. Генерация отчета в стандартном XML-формате

- *Возможность последующего автоматического анализа*
- *Публикация на web-страницах проекта*

Google Test

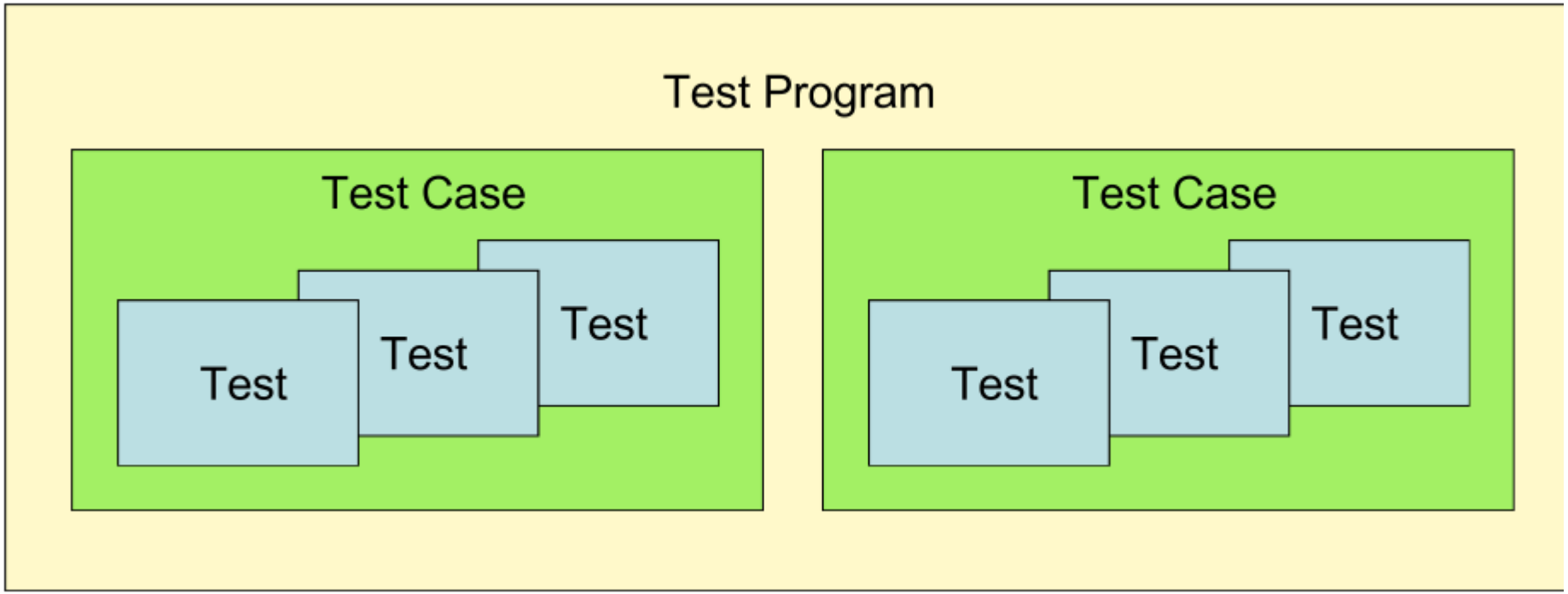
1. Популярный фреймворк для написания модульных тестов на C++, разработанный Google.
2. Open-source проект с BSD-лицензией (допускает использование в закрытых коммерческих проектах).
3. Используется в целом ряде крупных проектов
    - *Chromium, LLVM компилятор, OpenCV*
4. Написан на C++, строится при помощи CMake
    - *Поддерживает: Linux, Mac OS X, Windows, Cygwin, Windows CE, и Symbian*

5. Как правило используется в консольном режиме, но существует вспомогательное GUI приложение.

Возможности Google Test

- Automatic test discovery
- Rich set of assertions, user-defined assertions
- Death tests
- Fatal and non-fatal failures
- Value- and type-parameterized tests
- Various options for running the tests
- XML test report generation

Базовые концепции



- Каждый тест реализован как функция, с использованием макроса `TEST()` или `TEST_F()`.

- TEST() не только определяет, но и "регистрирует" тест.

Пример 1

```
#include <gtest/gtest.h>

TEST(MathTest, TwoPlusTwoEqualsFour) {
  EXPECT_EQ(2 + 2, 4);
}
```

Пример 2

Функция

```
int Factorial(int n); // Returns the factorial of n
```

Тесты

```
// Tests factorial of 0.
TEST(FactorialTest, HandlesZeroInput) {
  EXPECT_EQ(1, Factorial(0));
}

// Tests factorial of positive numbers.
TEST(FactorialTest, HandlesPositiveInput) {
  EXPECT_EQ(1, Factorial(1));
  EXPECT_EQ(2, Factorial(2));
  EXPECT_EQ(6, Factorial(3));
```

```
  EXPECT_EQ(40320, Factorial(8));
}
```

Пример 3

```
#include <gtest/gtest.h>
#include <vector>

using namespace std;

// A new one of these is created for each test
class VectorTest : public testing::Test {
public:
  vector<int> m_vector;

  virtual void SetUp() {
    m_vector.push_back(1);
    m_vector.push_back(2);
  }

  virtual void TearDown() {}
};

TEST_F(VectorTest, testElementZeroIsOne) {
  EXPECT_EQ(m_vector[0], 1);
}
```

```
TEST_F(VectorTest, testElementOneIsTwo) {
  EXPECT_EQ(m_vector[1], 2);
}

TEST_F(VectorTest, testSizeIsTwo) {
  EXPECT_EQ(m_vector.size(), (unsigned int)2);
}
```

Консольный лог Google Test

```
[mlong@n6-ws2 x86]$ bin/hellotest
Running main() from gtest_main.cc
[==========] Running 4 tests from 2 test cases.
[----------] Global test environment set-up.
[----------] 3 tests from VectorTest
[ RUN      ] VectorTest.testElementZeroIsOne
[       OK ] VectorTest.testElementZeroIsOne (0 ms)
[ RUN      ] VectorTest.testElementOneIsTwo
[       OK ] VectorTest.testElementOneIsTwo (0 ms)
[ RUN      ] VectorTest.testSizeIsTwo
[       OK ] VectorTest.testSizeIsTwo (0 ms)
[----------] 3 tests from VectorTest (0 ms total)

[----------] 1 test from MathTest
[ RUN      ] MathTest.Zero
[       OK ] MathTest.Zero (0 ms)
[----------] 1 test from MathTest (0 ms total)

[----------] Global test environment tear-down
[==========] 4 tests from 2 test cases ran. (0 ms total)
[  PASSED  ] 4 tests.
```

Полезные советы

Тесты можно временно выключать

```
TEST(MathTest, DISABLED_two_plus_two_equals_four)
{
  int x = 2 + 2;

  EXPECT_EQ(4, x);
}
```

Тесты можно фильтровать по имени при запуске

```
$ ./bin/hellotest --gtest_filter=*Vector*
```

У Google Test есть ряд других полезных опций

```
$ ./bin/hellotest --help
```

Юнит-тест курильщика

Юнит-т

```
[Test]
public void TestMethod1()
{
   var calc = new Calculator();
   calc.ValidOperation = Calculator.Operation.Multiply;
   calc.ValidType = typeof (int);
   var result = calc.Multiply(-1, 3);
   Assert.AreEqual(result, -3);
   calc.ValidOperation = Calculator.Operation.Multiply;
   calc.ValidType = typeof (int);
   result = calc.Multiply(1, 3);
   Assert.IsTrue(result == 3);
   if (calc.ValidOperation == Calculator.Operation.Invalid)
   {
      throw new Exception("Operation should be valid");
   }
   calc.ValidOperation = Calculator.Operation.Multiply;
   calc.ValidType = typeof (int);
   result = calc.Multiply(10, 3);
```

```
[TestMetho
public voi
    var pa

    page.A

    Assert
}
```

Авторство: Антон Бевзюк, SmartStepGroup.

Критерии хорошего теста

1. Короткий (имеет чистый код)
2. Сфокусированный (только один assert)
3. Быстрый
4. Автоматический
5. Независим от порядка исполнения и окружения

Паттерн AAA: Arrange, Act, Assert

Необходимость автоматических тестов

- Оптимизация производительности!
- Внесение изменений в уже отлаженные компоненты
- Коллективное владение
- Работа с унаследованым и сторонним кодом
- Портирование ПО на новые платформы
- Тестирование новых платформ

Современная стратегия тестирования

- Без "зеленых" тестов нет уверенности в работоспособности кода
- Фокус на максимальную автоматизацию
  - *Полное тестирование требуется несколько раз в день, каждому члену команды*
- **Тесты пишутся самими разработчиками, одновременно с реализацией**
  - *Тесты это лучшая документация, которая всегда актуальна (компилятор!)*

- *Тесты это первые сэмплы, показывающие простые примеры использования*
- *Test-Driven Development*

- Код тестируется **непрерывно**
  - *Это делается локально на машине разработчика*
  - *Это делается на сервере до того, как добавить его в репозиторий*

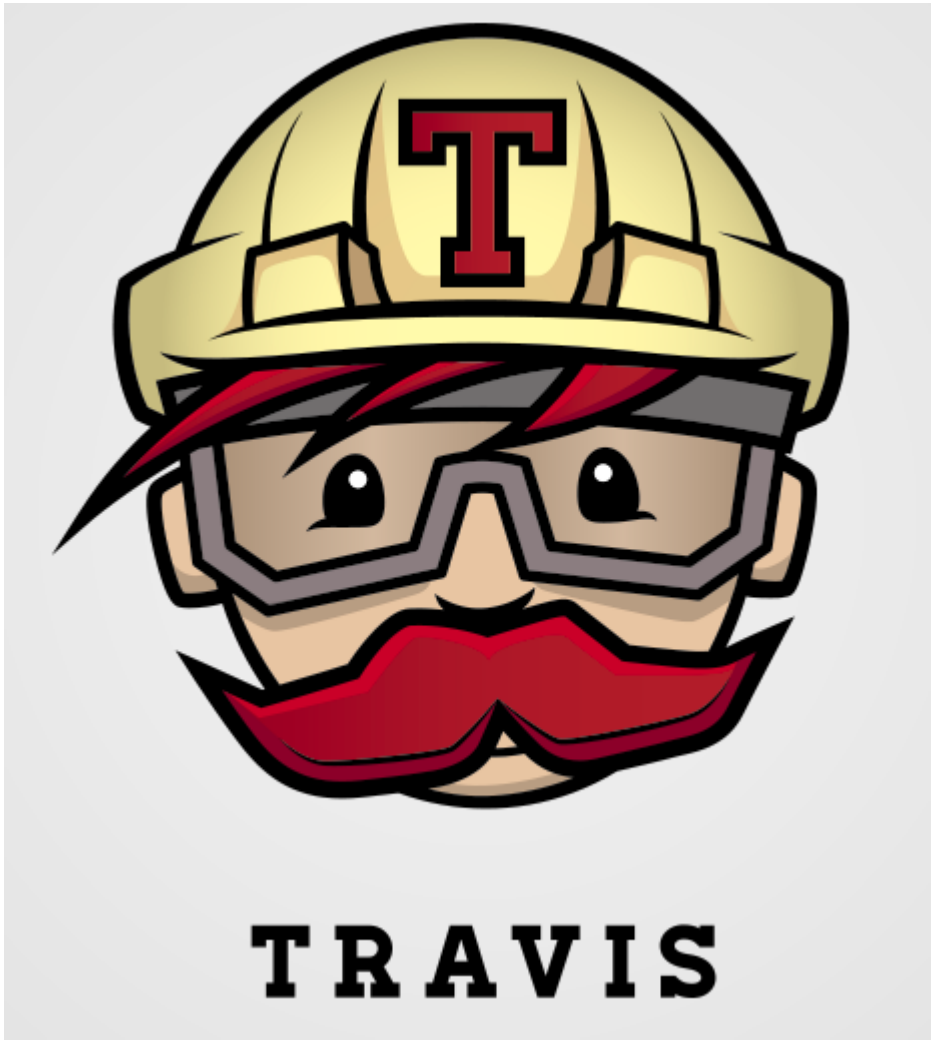
Современная стратегия тестирования (2)

- Автоматические тесты замещают отладку
  - *Предсказуемость времени разработки*
  - ***Пойманный баг документируется в виде теста***
- Тесты — это "first-class citizens"
  - *Стоит отдавать код вместе с тестами*
  - *Нужно заботиться о качестве кода тестов*
  - *Метафора тестов: скелет, позволяющий организму двигаться*

Google Test: Резюме

1. Пишите "жесткие" тесты, старайтесь сломать свой код.
2. Создание тестов — это составляющая самого процесса программирования.
    - *Почитайте про Test-Driven Development*
3. Без тестов нет уверенности в работоспособности кода.
4. Весь продуктовый код должен быть покрыт автоматическими тестами.
5. Автоматические тесты должны прогоняться при каждом изменении кода.

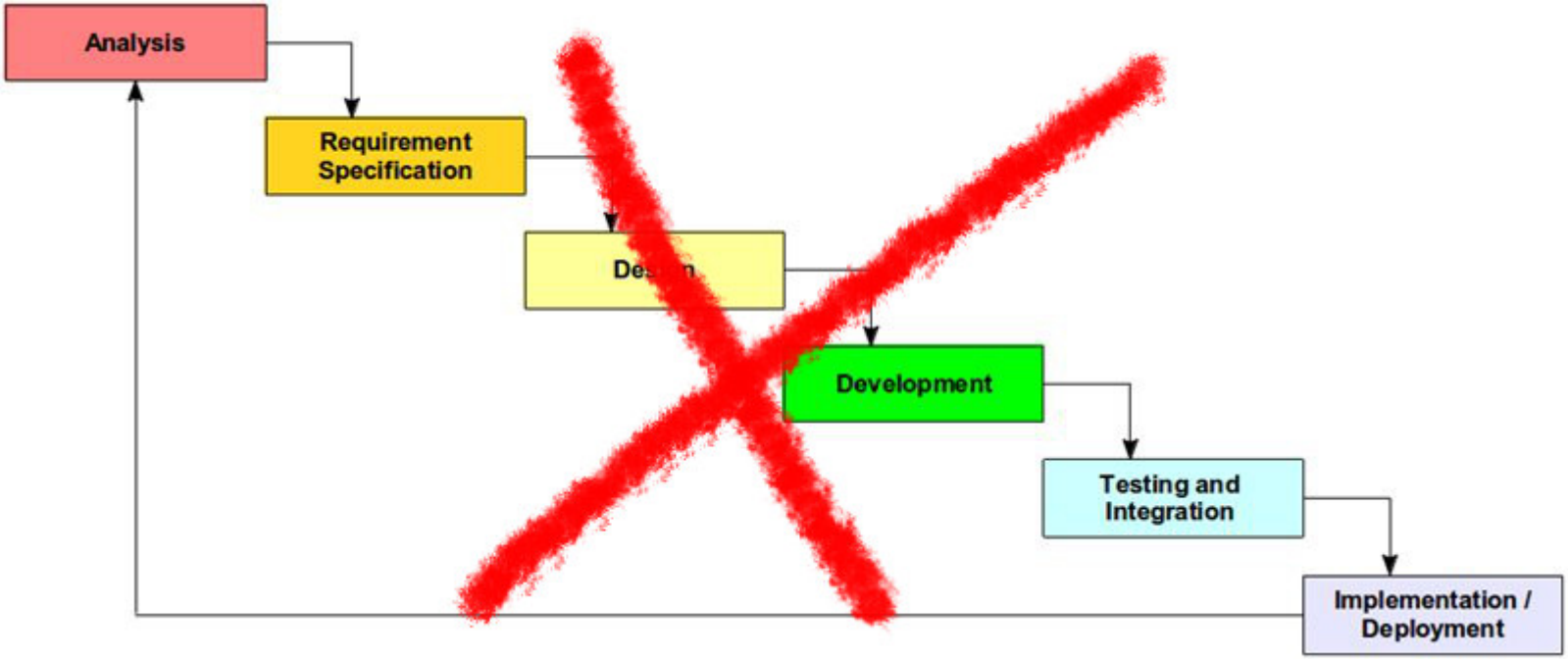
6. Ежедневно должно проводиться полное тестирование проекта,
   желательно с публикацией тестовых дистрибутивов (nightly builds).


T
TRAVIS

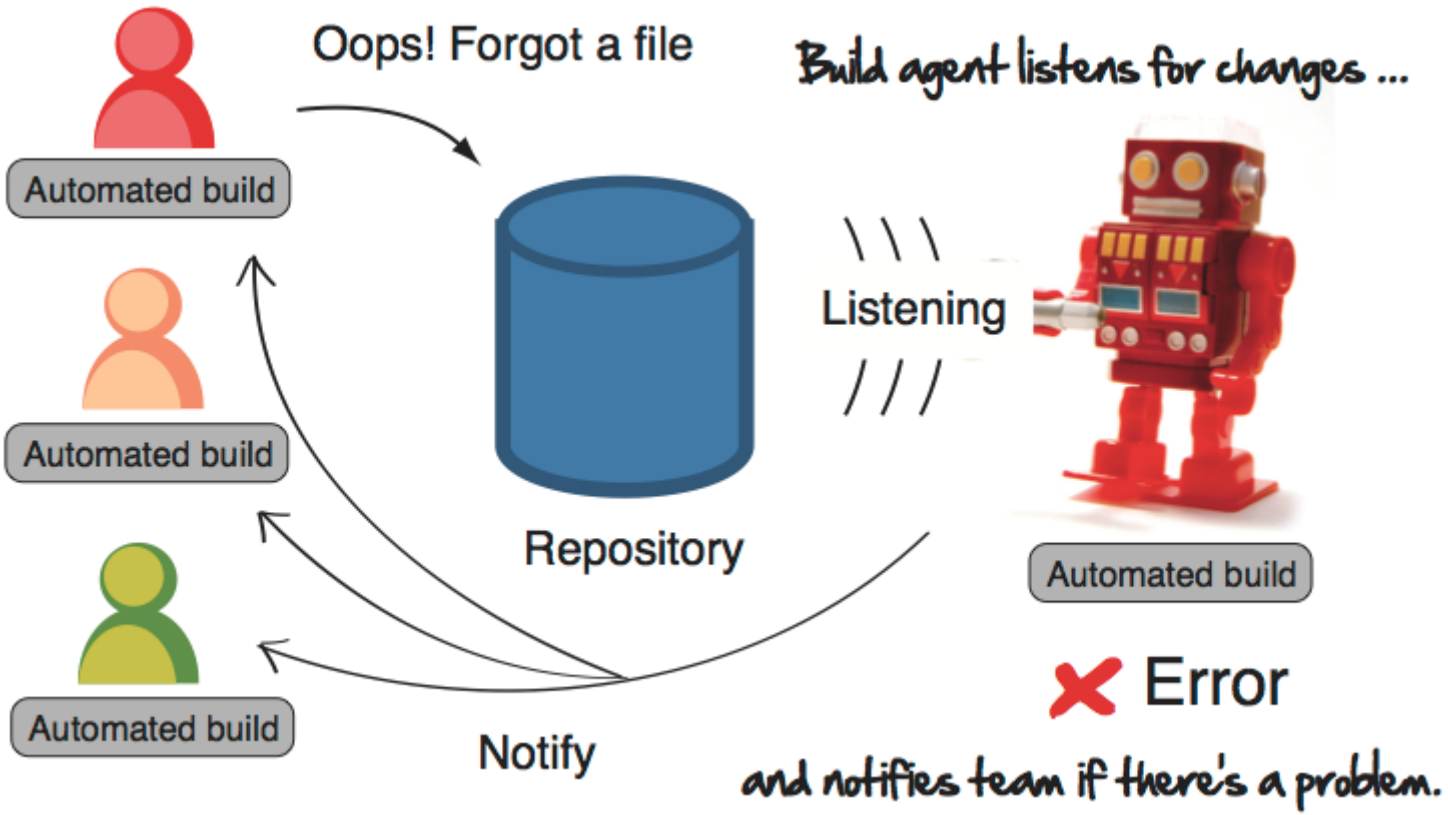
Непрерывная интеграция

Непрерывная интеграция (англ. Continuous Integration) *— это практика разработки ПО, которая заключается в выполнении* ***частых автоматизированных*** *сборок проекта для скорейшего выявления и решения интеграционных проблем.*

Непрерывная сборка — это **сердцебиение** вашего проекта.


Analysis
Requirement Specification
Development
Testing and Integration
Implementation / Deployment

Практика непрерывной интеграции



1. Все хранится в **централизованном репозитории**: исходный код, конфигурационные файлы, тестовые данные, сборочные и тестовые скрипты

2. **Полная автоматизация** операций с кодом: выкачивание из репозитория, сборка, тестирование и так далее.

Travis CI

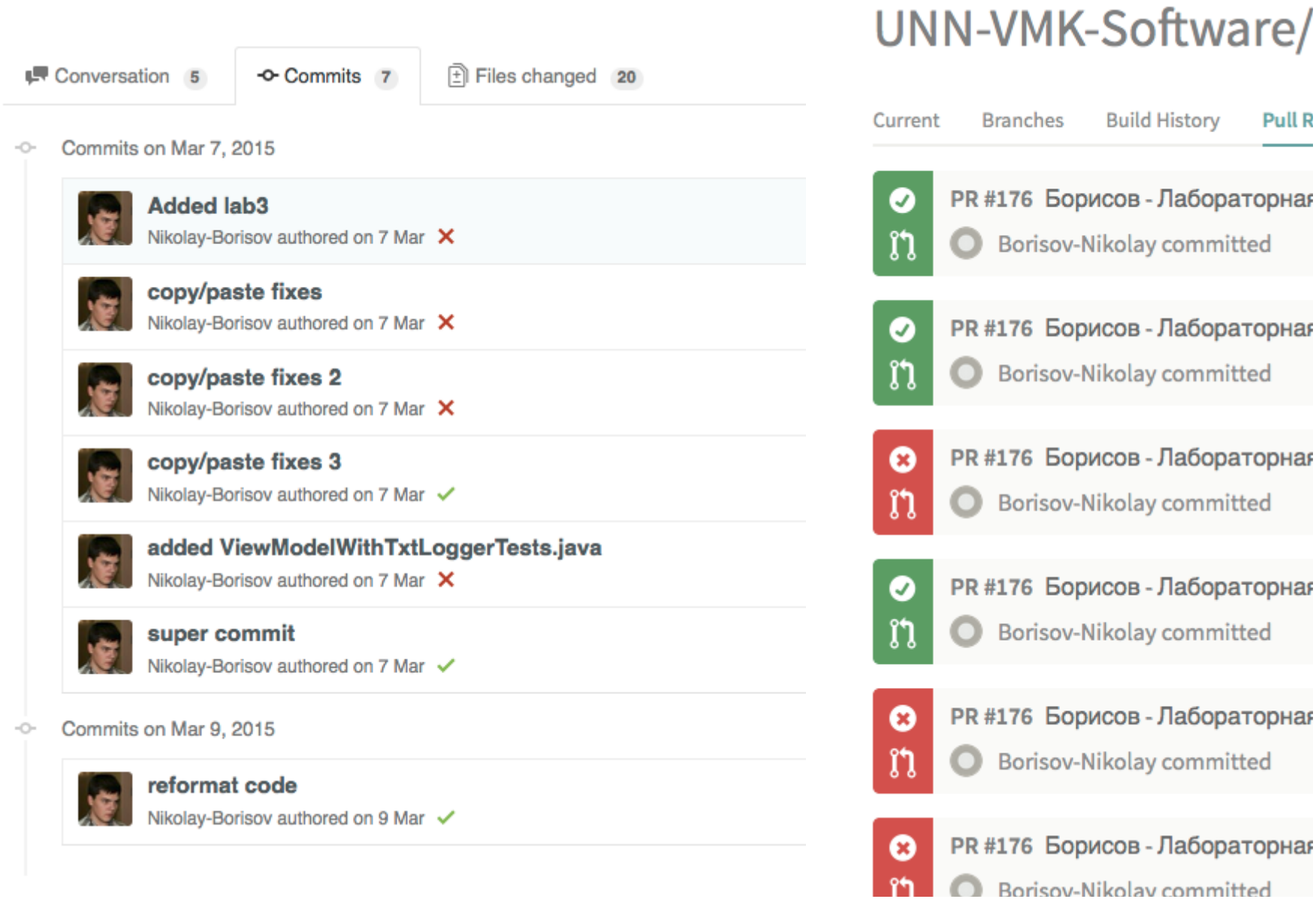
- Официальный сайт проекта: http://travis-ci.org
- Веб-сервис для сборки и тестирования ПО (open-source)
- Важными особенностями являются интеграция с GitHub и возможность

бесплатного использования

- Поддерживает большое количество языков: C, C++, Clojure, Erlang, Go, Groovy, Haskell, Java, JavaScript, Perl, PHP, Python, Ruby и Scala
- Тестирование

происходит на виртуальных Linux-машинах, запускаемых в облаке Amazon

GitHub + Travis CI


Conversation 5
Commits 7
Files changed 20
Commits on Mar 7, 2015
Added lab3
Nikolay-Borisov authored on 7 Mar
copy/paste fixes
Nikolay-Borisov authored on 7 Mar
copy/paste fixes 2
Nikolay-Borisov authored on 7 Mar
copy/paste fixes 3
Nikolay-Borisov authored on 7 Mar
added ViewModelWithTxtLoggerTests.java
Nikolay-Borisov authored on 7 Mar
super commit
Nikolay-Borisov authored on 7 Mar
Commits on Mar 9, 2015
reformat code
Nikolay-Borisov authored on 9 Mar
UNN-VMK-Software/
Current
Branches
Build History
Pull R
PR #176 Борисов - Лабораторная
Borisov-Nikolay committed
PR #176 Борисов - Лабораторная
Borisov-Nikolay committed
PR #176 Борисов - Лабораторная
Borisov-Nikolay committed
PR #176 Борисов - Лабораторная
Borisov-Nikolay committed
PR #176 Борисов - Лабораторная
Borisov-Nikolay committed
PR #176 Борисов - Лабораторная
Borisov-Nikolay committed

Современная стратегия тестирования

- Без "зеленых" тестов нет уверенности в работоспособности кода
- Фокус на максимальную автоматизацию
  - *Полное тестирование требуется несколько раз в день, каждому члену команды*
- **Тесты пишутся самими разработчиками, одновременно с реализацией**
  - *Тесты это лучшая документация, которая всегда актуальна (компилятор!)*

- *Тесты это первые сэмплы, показывающие простые примеры использования*
- *Test-Driven Development*

- Код тестируется **непрерывно**
  - *Это делается локально на машине разработчика*
  - *Это делается на сервере до того, как добавить его в репозиторий*

Современная стратегия тестирования (2)

- Автоматические тесты замещают отладку
  - *Предсказуемость времени разработки*
  - ***Пойманный баг документируется в виде теста***
- Тесты — это "first-class citizens"
  - *Стоит отдавать код вместе с тестами*
  - *Нужно заботиться о качестве кода тестов*
  - *Метафора тестов: скелет, позволяющий организму двигаться*

Резюме

В рамках нашей школы будут выполняться *учебные* задания, однако они будут носить все черты *промышленной* разработки ПО.

- Кросс-платформенность
  - *Построение с использованием CMake*
  - *Тестирование на Linux (gcc, clang)*
  - *Потенциальная переносимость на Android и iOS*
- Коллективная разработка
  - *Использование Git*

- *Peer code review от преподавателей и коллег средствами GitHub*

- Тестирование
  - *Автоматические тесты на базе Google Test*
  - *Непрерывная интеграция на основе Travis-CI*

Ссылки

1. Wikipedia "Системы контроля версий".
2. Pro Git by Scott Chacon.
3. "Mercurial tutorial" by Joel Spolsky.
4. GTest
5. Google Test Talk

Спасибо!

Вопросы?