Обзор доступных Open Source-инструментов для решения всех возникающих при разработке ПО задач

Стандартный набор приложений

- Создавать проект программного продукта: набор пакетов и классов и их взаимодействие; взаимодействие с агентом/сервером и с пользователем.
- Писать код программы.
- Тестировать код программы.
- Отлаживать код.
- Писать документацию на протестированный и отлаженный код.
- Хранить код в едином, общедоступном хранилище.
- Писать инструкцию по использованию программного продукта.
- Создавать пакет развертывания.
- Собирать информацию об ошибках в программном

Проект программного продукта. UML-редактор

NetBeans IDE (http://www.netbeans.org/)

- написан на Java
- все виды диаграмм UML
- не умеет генерировать исходный код заголовочных файлов на языке C++, а только прототипы классов для Java.

ArgoUML (http://argouml.tigris.org/)

- написан на Java
- все виды диаграмм UML
- проектирование структуры таблиц для баз данных.

StarUML (http://staruml.sourceforge.net/)

- все виды диаграмм.
- проект не развивается с 2005 года.

Umbrello (http://uml.sourceforge.net/)

- все виды диаграмм.
- генерирует код для языков C++ и Java.

Среда разработки(IDE)

Eclipse (http://www.eclipse.org/)

- разработки программ на нескольких языках программирования.
- автодополнение кода
- история изменений
- проекты
- системамы контроля версий CVS и SVN
- расширения/плагины
- взаимодействие с отладчиком.

NetBeans (http://www.netbeans.org/)

- поддерживает множество языков программирования
- автодополнение кода
- проекты
- системамы контроля версий CVS и SVN
- расширения
- встроенный UML-редактор
- взаимодействие с отладчиком.

QtCreator (https://www.qt.io/ide/)

- C/C++, Qml.
- автодоплонение кода
- проекты
- системы контроля версий Git и SVN
- плагины, расширения.
- взаимодействие с отладчиком
- рефакторинг

Тестирование функционала

- CppUnit
- Boost
- **CppTest**
- QTest

Статический анализатор кода

- CppCat: все диагностики (нет уровней);
- **Cppcheck**: Errors и Warnings;
- PVS-Studio: 1 и 2 уровень диагностик общего назначения;

Документирование

- Doxygen
- QDoc

Сбор информации об ошибках

- Bugzilla
- Redmine

GNU Prof — анализатор производительности

Jenkins — Система непрывной интеграции

Системы контроля версий

Проблемы:

- пишите код, статью совместно с соавтором(совместная разработка, crowdsourcing)
- необходимость одновременной правки (2-3чел.)
- Новые правки не должны пропадать
- В режиме нон-стоп

Разработка ПО:

- Некая версия 2.0 замечательной Инф. Системы
- Версия 2.0 сырая и не собирается
- Заказчик находит критический баг в предыдущей версии
- Кто добавил код(добавили бажный код)

Управление конфигурациями(software configuration manager):

- Программные продукты
- Информационные модели и иные артефакты
- Железо
- Средства аудита самой системы
- Метрики работы системы(журналы, архивы и тд.)

Системы контроля версий это лишь частный случай Управление конфигурациями:

- Архивирование
- Документирование
- Учёт ошибок(бактрекеры)
- Управление сборкой
- Учёт окружения
- Поддержка процессов разработки

Система управления версиями (Version Control System, VCS или *Revision Control System):*

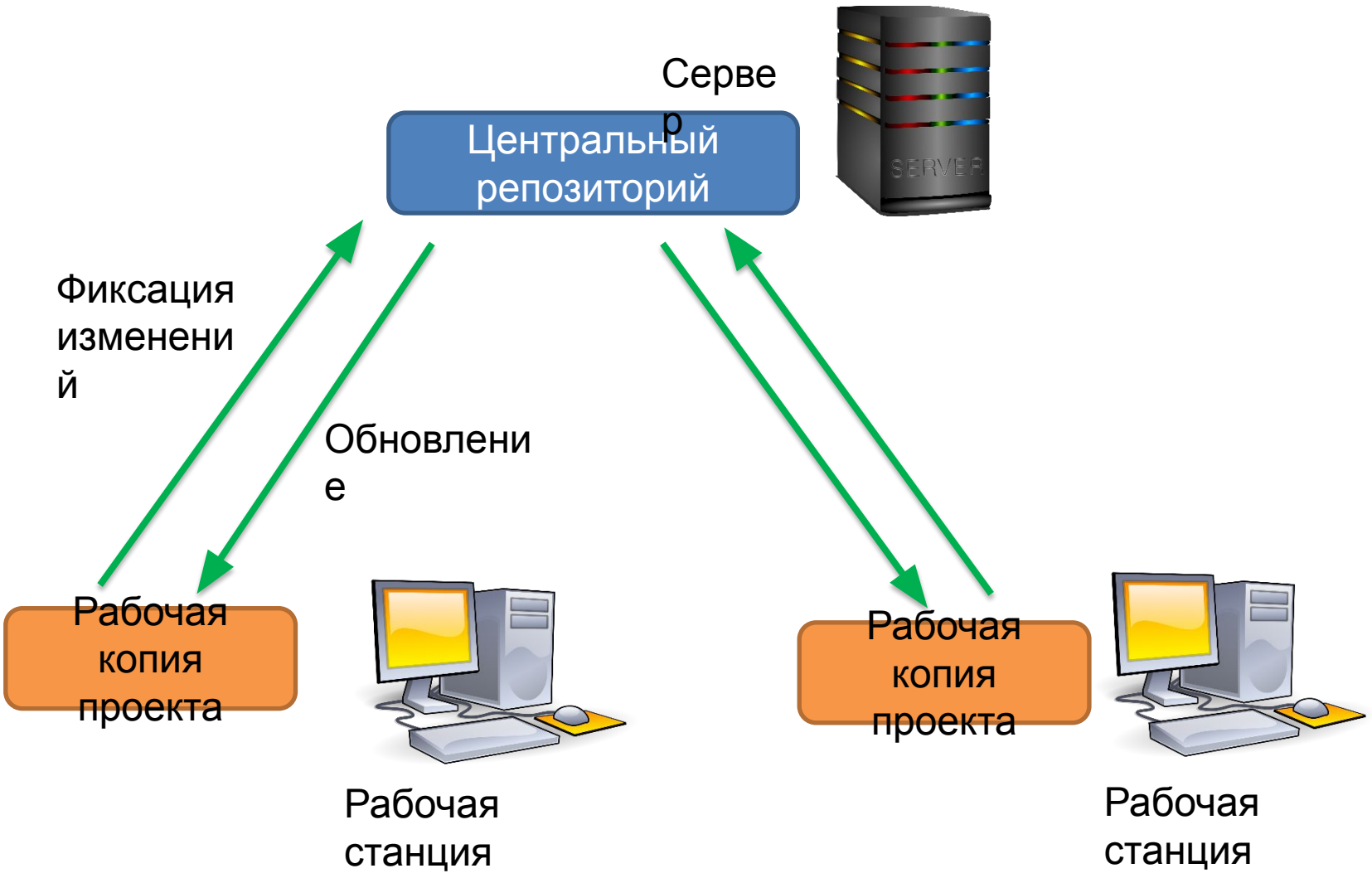
- Хранение версий программных артефактов(коды, документы и документации в целом, модели САПР)
- Мгновенный доступ к любой версии
- Обеспечение совместной работы команды.

Типы:

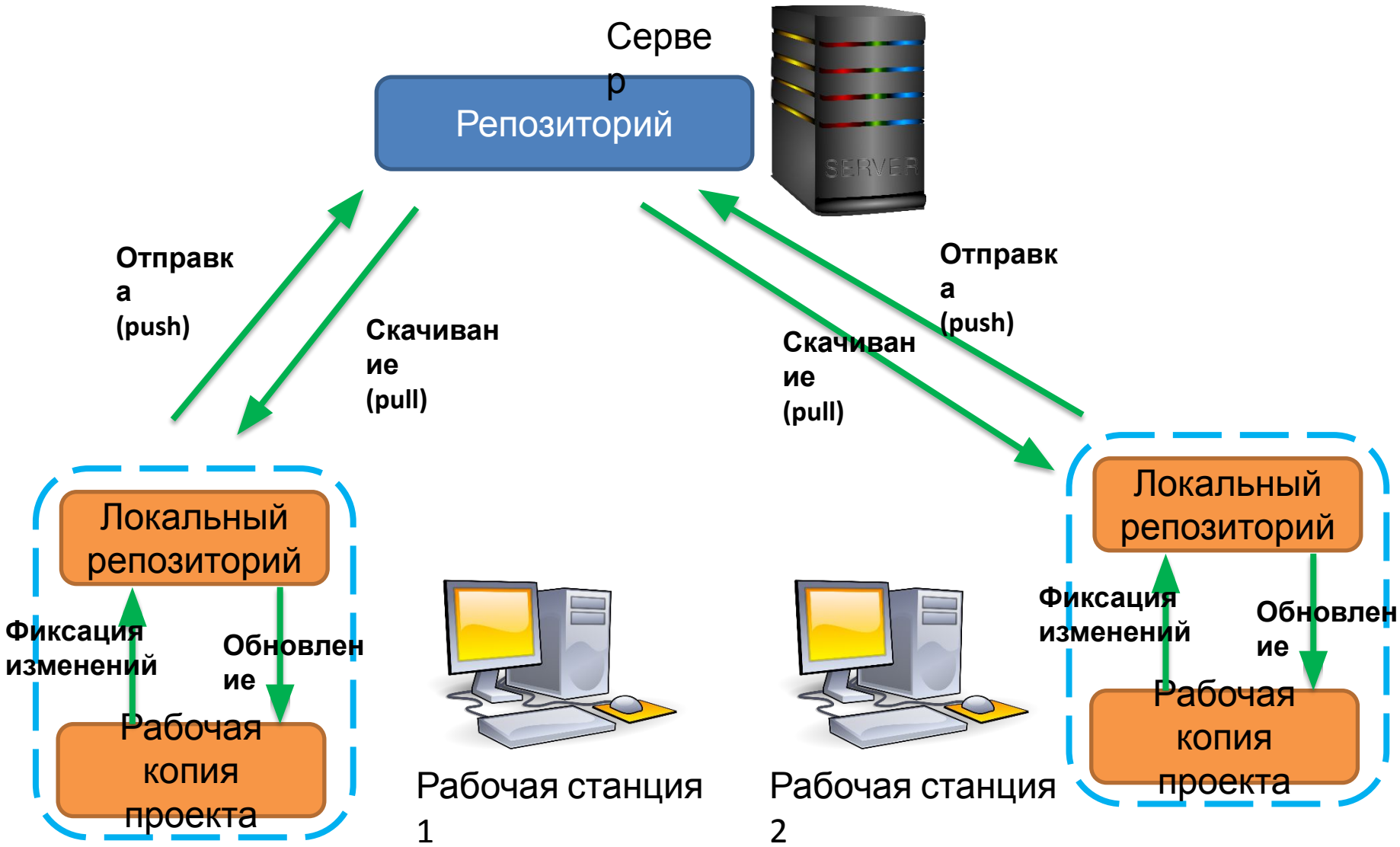
- Однопользовательские (Локальные, Файловые системы)
- Многопользовательские:
  - Централизованные(**Subversion, CVS**)
  - Распределённые(**Git**, **Mercurial**, **Bazaar**)

Repository	Хранилище артефактов(документов файлов)
Revision	Версия документа.
Workspace	Рабочая\локльная копия документов

Централизованные VCS



Распределённые VCS

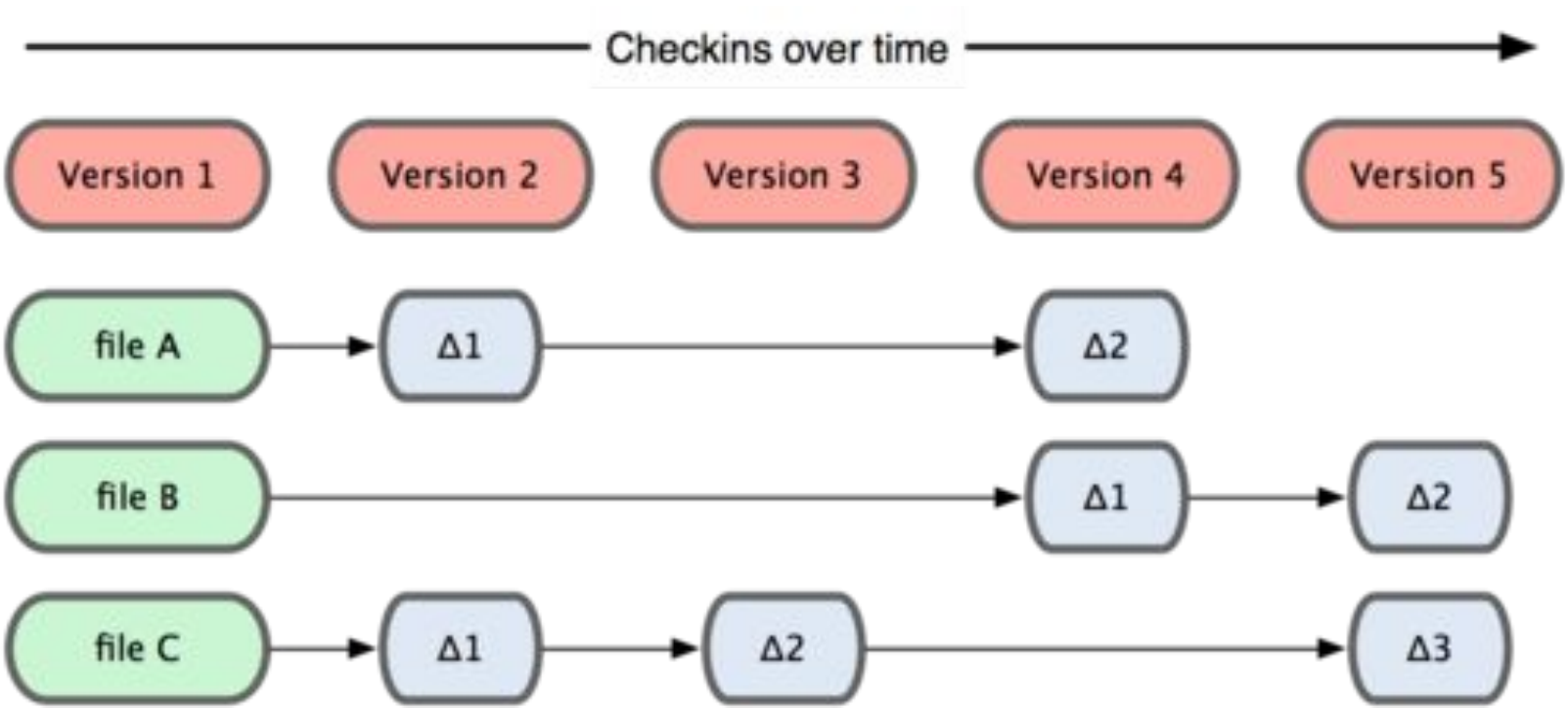


- *1970 make система сборки*
- *1972 первая система контроля версий SCCS(source code contr.sys)*
- *1980 RCS (Revis. Control system) утилита patch*
- *1986 CVS (Concurrent Version System) первая многопользовательская Система контроля версий*
- *2000 CollabNet начало работ над «Subversion»*
- *200х появились распределённые Системы контроля версий*
- *2009 год. **GIT**, Mercurial, Bazaar*

SVN (централизованная)

- В CVS каталог не отслеживался как объект
- Транзакции. Упал коннект на сервер ушли фрагментированные изменения
- Модификации имён файлов (удаление –создание но потеря истории изменений)
- Метаданные
- Блокировки(работа с бинарным файлом)
- Эффективность клиент-серверного обмена(хранение 2х версий текущ. и преды-отсыл изменений а не всего коммита)
- Эффективность хранения двоичных файлов
- Эффективность создания ветвей и меток
- Эффективность использования памяти сервера

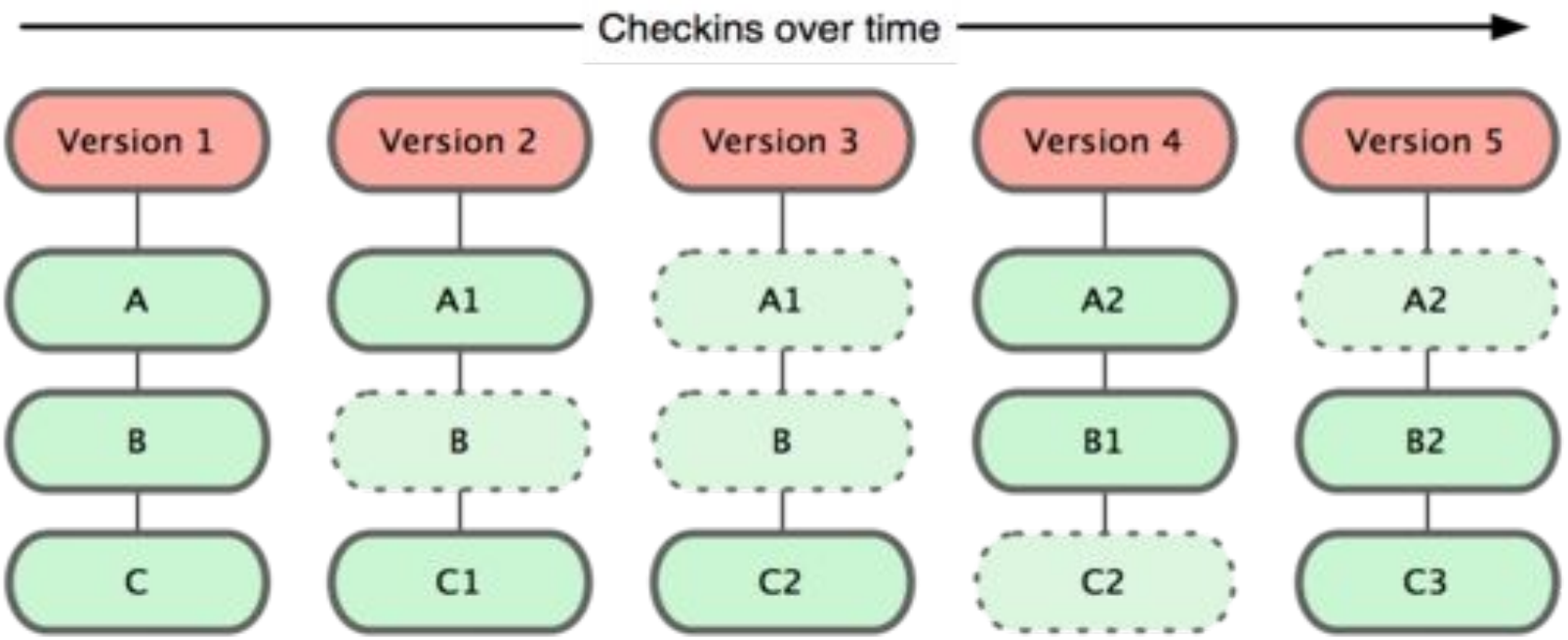
Другие системы хранят данные как изменения к базовой версии для каждого файла



Достоинство:
Экономия места

Недостаток:
Увеличенное время вычисления состояния файлов

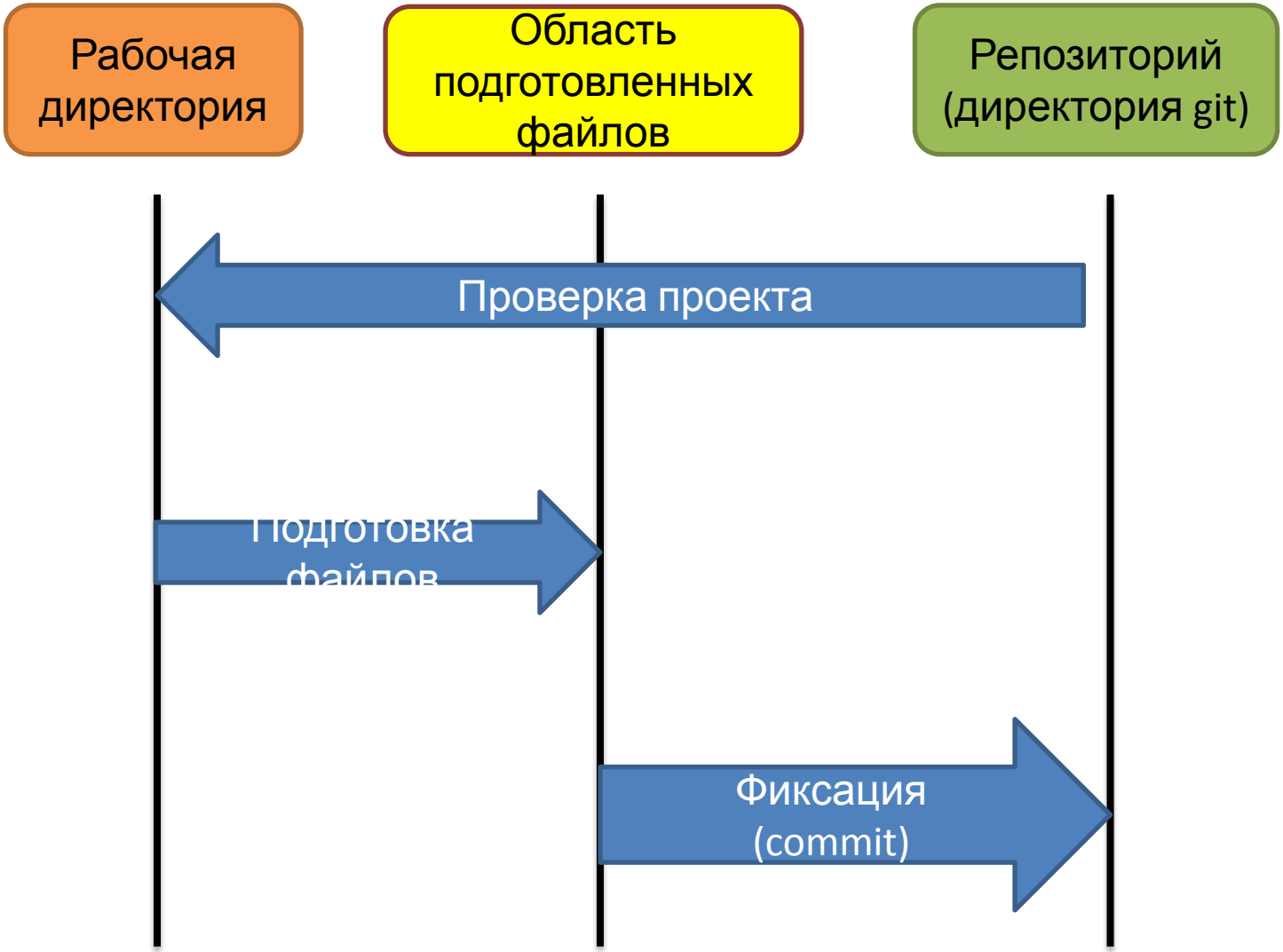
Git хранит данные как слепки состояний проекта во времени



Достоинство:
Быстрый доступ к ревизиям

Недостаток:
Увеличение объёма хранимой информации

Локальные операции



Начало

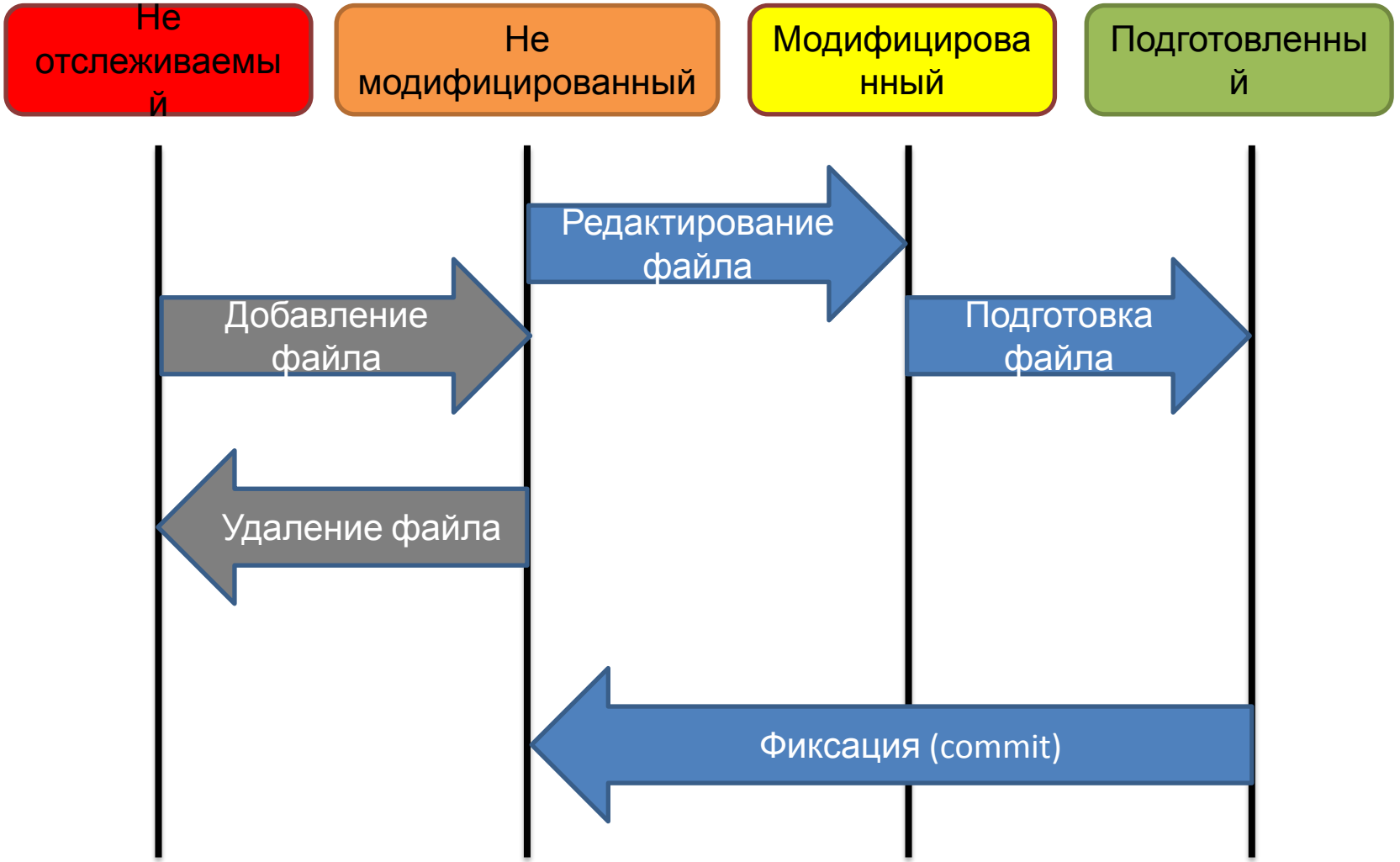
Создание репозитория в существующем каталоге

git init

Клонирование существующего репозитория

git clone git://github.com/schacon/grit.git

Жизненный цикл файлов в git



Определение состояния файлов

$ git status

 # On branch master
nothing to commit (working directory clean)

Внесение изменений в файл

$ vim README
$ git status

 # On branch master
 # Untracked files:
 #   (use "git add <file>..." to include in what will be committed)
#
 #   README
nothing added to commit but untracked files present (use "git add" to track)

Отслеживание новых файлов

```
$ git add README

$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#   new file:   README
#
```

Индексация измененных файлов

```
$ git status

# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   README
#
# Changed but not updated:
#   (use "git add <file>..." to update what will
be committed)
#
#       modified:   benchmarks.cpp
```

```
$ git add benchmarks.cpp
$ git status

# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   README
#       modified:   benchmarks.cpp
```

Изменение в не зафиксированном файле

$ vim benchmarks.rb

$ git status

 # On branch master
 # Changes to be committed:
 #   (use "git reset HEAD <file>..." to unstage)
#
 #   new file: README
 #   modified: benchmarks.cpp
#
 # Changed but not updated:
 #   (use "git add <file>..." to update what will be committed)
#
 #   modified: benchmarks.cpp
#

$ git add benchmarks.rb

$ git status

 # On branch master
 # Changes to be committed:
 #   (use "git reset HEAD <file>..." to unstage)
#
 #   new file: README
 #   modified: benchmarks.cpp
#

Фиксация изменений

$ git commit -m "Story 182: Fix benchmarks for speed “

[master]: created 463dc4f: "Fix benchmarks for speed"
2 files changed, 3 insertions(+), 0 deletions(-) create mode 100644 README

Удаление файлов

$ git rm grit.gemspec
$ git status

 # On branch master
#
 # Changed but not updated:
 #   (use "git add/rm <file>..." to update what will be committed)
#
 #   deleted: grit.gemspec#

Перемещение файлов

$ git mv file_from file_to

Просмотр истории коммитов

$ git log

commit ca82a6dff817ec66f44342007202690a93763949
Author: Scott Chacon <schacon@gee-mail.com>
Date: Mon Mar 17 21:52:11 2008 -0700
changed the version number

*commit **085b**b3bcb608e1e8451d4b2432f8ecbe6306e7e7*
Author: Scott Chacon <schacon@gee-mail.com>
Date: Sat Mar 15 16:40:33 2008 -0700
removed unnecessary test code

commit a11bef06a3f659402fe7563abf99ad00de2209e6
Author: Scott Chacon <schacon@gee-mail.com>
Date: Sat Mar 15 10:31:28 2008 -0700
first commit

*$ git checkout **085b***

Отмена изменений

Отмена локальных изменений файла

$ git checkout master

Файл: изменён, **НЕ** проиндексирован, **НЕ** произведён коммит

Отмена индексации файла

Файл: изменён, проиндексирован, **НЕ** произведён коммит

$ git add .

$ git status

 # On branch master
 # Changes to be committed:
 #   (use "git reset HEAD <file>..." to unstage)
#
 #   modified: README.txt
 #   modified: benchmarks.cpp

$ **git reset HEAD benchmarks.cpp**
# benchmarks.cpp: locally modified

$ **git status**
 # On branch master
 # Changes to be committed:
 #   (use "git reset HEAD <file>..." to unstage)
#
 #   modified: README.txt
#
 # Changes not staged for commit:
 #   (use "git add <file>..." …)
 #   (use "git checkout -- <file>...")
#
 #   modified: benchmarks.cpp

Изменение(перезапись) последнего коммита

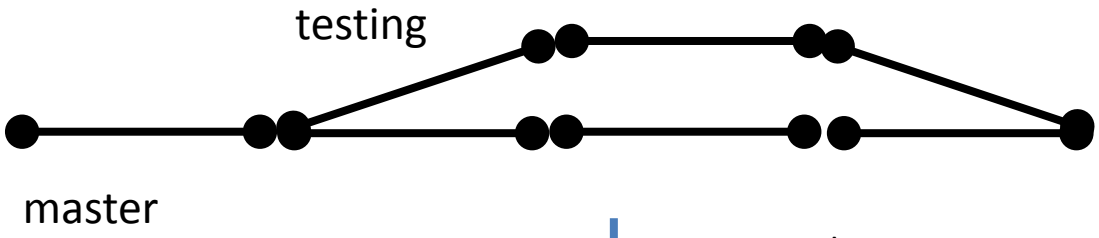
$ git commit -m 'initial commit'
$ git add forgotten_file
$ **git commit --amend**

Отмена коммитов

$ git revert HEAD

$ git hist
* 45fa96b 2011-03-09 | Revert "Oops, we didn't want this commit" (HEAD, master)
* 846b90c 2011-03-09 | Oops, we didn't want this commit [Alexander Shvets]
* fa3c141 2011-03-09 | Added HTML header (v1) [Alexander Shvets]
* 8c32287 2011-03-09 | Added standard HTML page tags (v1-beta) [Alexander Shvets]
* 43628f7 2011-03-09 | Added h1 tag [Alexander Shvets]
* 911e8c9 2011-03-09 | First Commit [Alexander Shvets]

Ветвление в Git




```
$ git branch testing

$ git checkout testing

$ git checkout master
$ git merge testing
Updating f42c576..3a0874c
Fast forward
 README |    1 -
 1 files changed, 0 insertions(+), 1 deletions(-)
```

```
[master*]$ git status
index.html: needs merge
# On branch master
# Changed but not updated:
#   (use "git add <file>..." to update what
#   (use "git checkout -- <file>..." to discard
#
#       unmerged:   index.cpp
#
```

```
<<<<<<< HEAD: index.cpp
A=B+C;
=======
A=B-C;
>>>>>>> iss53:index.cpp
```

Работа с удалёнными репозиторями

$ git remote
origin

$ git remote add pb git://github.com/paulboone/ticgit.git

$ git remote -v
origin	git://github.com/schacon/ticgit.git
pb	git://github.com/paulboone/ticgit.git

Fetch и Pull

$ git fetch [remote-name] // синхронизация без сброса локальных изменений

$ git pull [remote-name] // синхронизация со сбросом локальных изм.

Отправка изменений на сервер

git push [удал. сервер] [ветка]

Прмер:
$ git push origin master