

Системы контроля версий. Документация проекта.

Елена Кантонистова

План на сегодня

- Системы контроля версий + практика
- Документация проекта
- Разведочный анализ данных и построение моделей машинного обучения + практика

Системы контроля версий

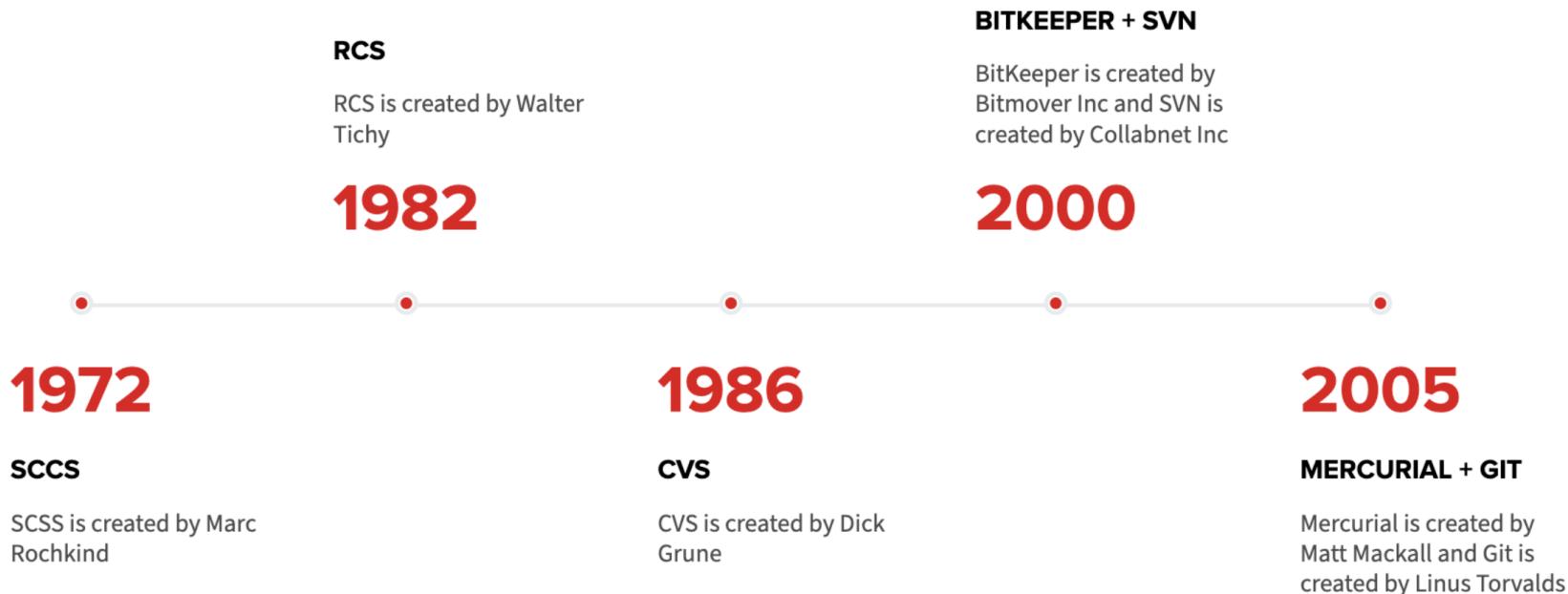
- Если вы работаете над проектом, тем более в составе команды, необходимо сохранять промежуточные этапы.

Система контроля версий – это система, записывающая изменения

в файл или набор файлов в течение времени и позволяющая вернуться позже к определенной версии.

Системы контроля версий

- **Локальные (RCS)**
- **Централизованные (CVS, Subversion и Perforce)**
- **Распределенные (Git, Mercurial, Bazaar или Darcs)**



Системы контроля версий

Локальные (RCS)

- для каждого файла, зарегистрированного в системе, хранится полная история изменений

Недостатки?

Системы контроля версий

Локальные (RCS)

- для каждого файла, зарегистрированного в системе, хранится полная история изменений

Недостатки?

- Не подходит для работы в команде

Системы контроля версий

Централизованные (CVS, Subversion и Perforce)

- Для организации такой системы контроля версий используется единственный сервер, который содержит все версии файлов
- Клиенты, обращаясь к этому серверу, получают из этого централизованного хранилища.

Недостатки?

Системы контроля версий

Централизованные (CVS, Subversion и Perforce)

- Для организации такой системы контроля версий используется единственный сервер, который содержит все версии файлов
- Клиенты, обращаясь к этому серверу, получают из этого централизованного хранилища.

Недостатки?

- Если сервер упал, то работа встанет

Системы контроля версий

Распределенные (Git, Mercurial, Bazaar или Darcs)

- клиенты не просто выгружают последние версии файлов, а полностью копируют весь репозиторий (место, где хранятся и поддерживаются какие-либо данные)
- можно выделить центральный репозиторий (условно), в который будут отправляться изменения из локальных и, с ним же эти локальные репозитории будут синхронизироваться

Git

Стандарт де-факто на сегодня – распределенная система контроля версий Git.



GitHub

GitHub – веб-сервис, который основан на системе Git.

- Это своего рода социальная сеть для разработчиков, которая помогает удобно вести коллективную разработку IT-проектов
- Здесь можно публиковать и редактировать свой код, комментировать и обсуждать его, следить за новостями других пользователей



Как работать с GitHub?

- **Через веб-интерфейс**
- Через командную строку
- Через графические оболочки

Наш рабочий вариант: через веб-интерфейс

При желании можете попробовать последний вариант.

GitHub – веб-интерфейс: скачать репозиторий

The screenshot shows a GitHub repository page for 'Murcha1990 / Raiff_practical_ML'. The 'Code' tab is selected. On the left, there's a file tree with files like 'Update Git.pdf', 'Lecture1_project_steps', 'Create readme.md', 'Lecture2_data', 'Create readme.md', 'Lecture3_git', 'Update Git.pdf', 'README.md', and 'Update README.md'. Below the file tree is a section for 'README.md'. On the right, there's an 'About' section with a description of the repository: 'Практический курс по машинному обучению для сотрудников Райффайзен банка.' (Practical course on machine learning for employees of Raiffeisen Bank). There are also sections for 'Readme', 'Stars' (0), 'Watching' (1), 'Forks' (0), and 'Releases' (none published). A prominent red oval highlights the 'Download ZIP' button under the 'Clone' section.

Murcha1990 / Raiff_practical_ML Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags

Go to file Add file ▾ Code ▾

Local Codespaces

Clone

HTTPS SSH GitHub CLI

git@github.com:Murcha1990/Raiff_practical_ML. ↗

Use a password-protected SSH key.

Open with GitHub Desktop

Download ZIP

About

Практический курс по машинному обучению для сотрудников Райффайзен банка.

Readme

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Raiff_practical_ML

GitHub – создать репозиторий

The screenshot shows a GitHub repository page for 'Murcha1990 / Raiff_practical_ML'. The 'Code' tab is selected. A context menu is open in the top right corner, with 'New repository' highlighted in blue. Other options in the menu include 'Import repository', 'New codespace', 'New gist', and 'New organization'. The repository page displays a commit history with three commits:

- Murcha1990 Update Git.pdf (7dae4e5, 3 hours ago, 7 commits)
- Lecture1_project_steps Create readme.md (yesterday)
- Lecture2_data Create readme.md (yesterday)

The 'About' section on the right provides a brief description of the repository: 'Практический курс по машинному обучению для сотрудников Райффайзен банка.'

При создании репозитория обязательно ставьте галочку “create README”!

GitHub Desktop

- Разработчики чаще всего работают с Git и GitHub через командную строку
- Есть и альтернативы – графические оболочки для Git/GitHub



GitHub
Desktop

GitHub Desktop – сервис с графическим интерфейсом для работы с Git/GitHub, заменяющий командную строку.

Скачать пакет можно с официального сайта <https://desktop.github.com/>

Основы работы с Git

В Git обширный функционал, но для знакомства с инструментом понадобится только несколько основных команд:

- **git clone** – клонирует существующий репозиторий на локальную машину
- **git commit** – добавляет файлы в список отслеживаемых для регистрации истории изменений
- **git push** – добавление локальных изменений в проект на GitHub
- **git pull** – загрузка изменений с GitHub на локальную машину

Репозиторий с материалами курса

- Изучаем репозиторий

https://github.com/Murcha1990/Raiff_PracticalML_May2023

- Скачиваем или клонируем себе репозиторий

Домашнее задание

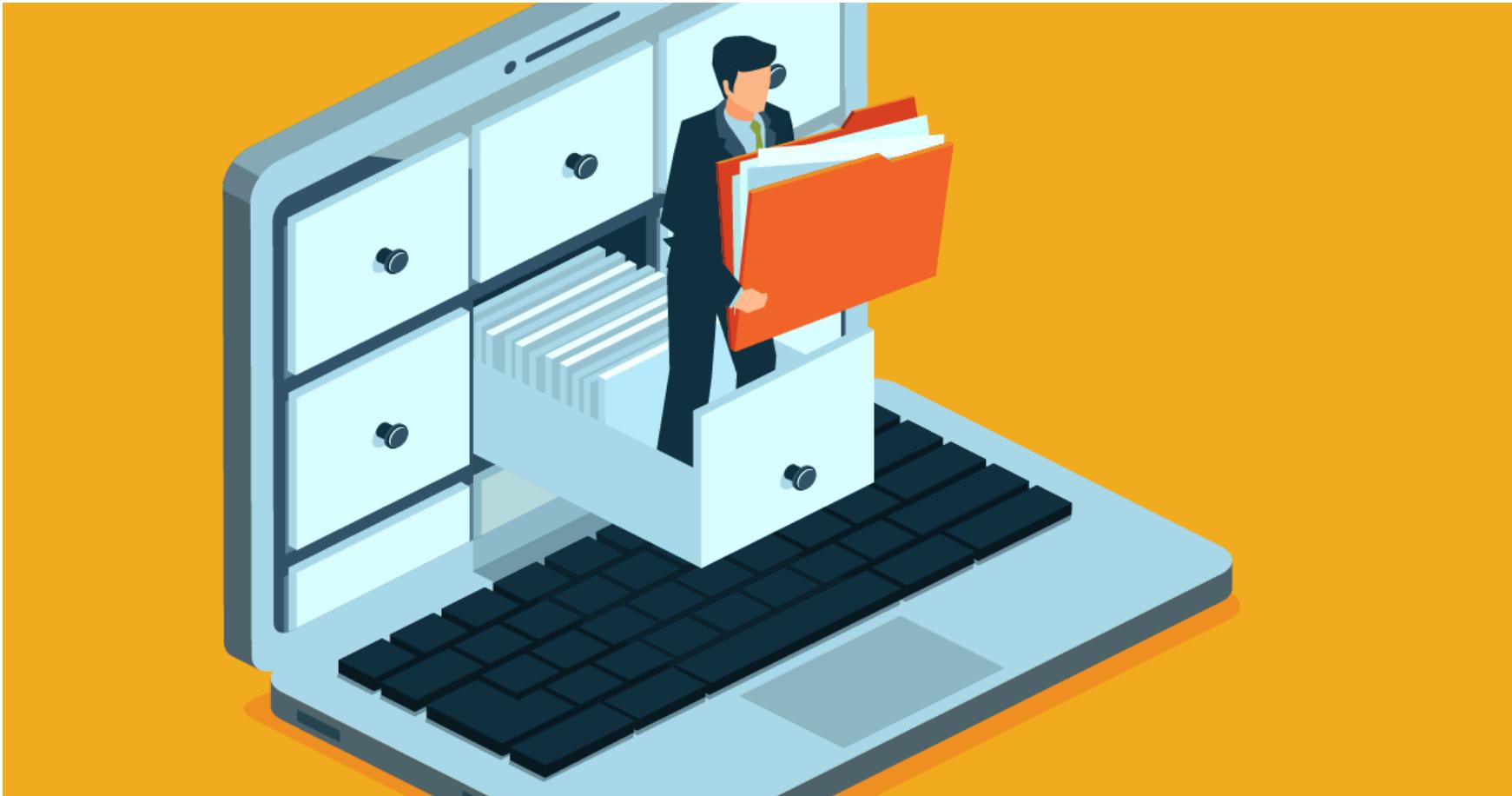
Шаг 1

- Создаем репозиторий командного проекта, называем осмысленно
- Заполняем README описанием задачи (копируем туда описание данных и самой задачи)
- Скачиваем репозиторий на локальный компьютер каждого из участников команды

Шаг 2

- Загружаем в папку на локальной машине файл с данными из БД
- Загружаем в папку на локальной машине ноутбук (ipynb-файл) с экспериментами
- Загружаем эти файлы на GitHub

Документация проекта



Документация проекта

Зачем нужна документация?

- **Если порядок не поддерживается - люди охотнее его нарушают и не следуют правилам**



Зачем писать?

Зачем нужна документация?

- Если порядок не поддерживается - люди охотнее его нарушают и не следуют правилам
- При уходе сотрудника, занимающегося проектом, при отсутствии документации знания о проекте уйдут вместе с ним
- При приходе нового сотрудника документация наиболее быстро и полно введет его в курс дела

Что писать?

- Техническое задание:
 - Сроки и общие требования к продукту
 - Описание разрабатываемого продукта:
 - используемые инструменты
 - интерфейс
- Технические подробности продукта (для разработчиков)
- Пользовательская документация

Что писать? Необходимый минимум

1. Документ-маршрутизатор

- место, откуда можно добраться до любой информации о проекте (это может быть документ с набором ссылок)



2. Артефакты проектных процессов

- статусы задач, план-график проекта, список нужных контактов и т.д.

3. Глоссарий

- терминология, используемая в проекте

Что писать? Необходимый минимум

4. Описание бизнес-целей и бизнес-процесса

- постановка задачи от заказчика со всеми необходимыми деталями

5. Концептуальная модель системы

- описание основных сущностей,

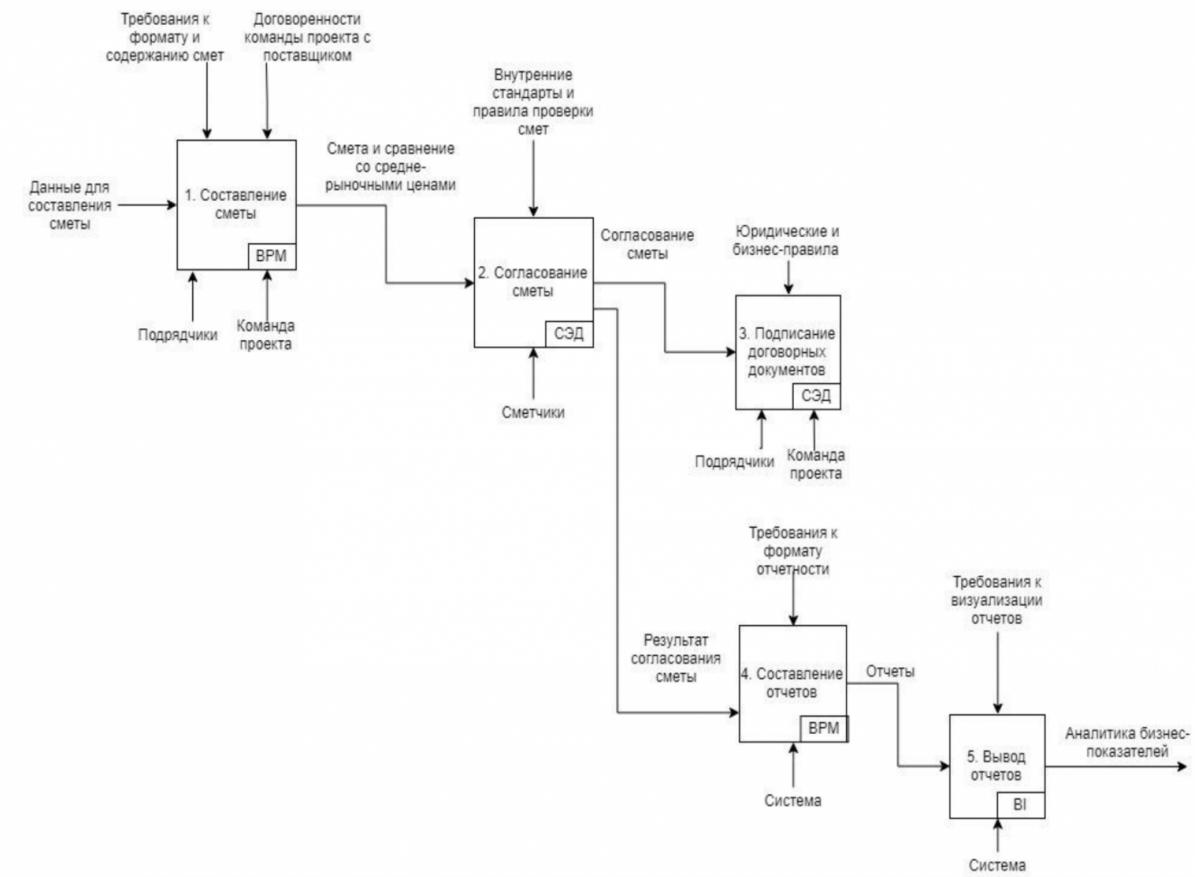
верхнеуровневая модель

функциональности,

взаимодействие с другими системами

6. Классы пользователей и уровни доступа

- памятка для кого и что делается



Что писать? Необходимый минимум

7. Сценарии использования продукта

- инструкция для пользователей, инструкция для тех.поддержки

8. Логика работы системы

- формулы расчета основных показателей

9. Описание API

- описание интерфейса разрабатываемого приложения

10. Тестовые данные

- среды, учетные записи

11. Ограничения/нефункциональные требования

- на сколько пользователей рассчитывать нагрузку? Как часто делается обновление данных и т.д.

Общие мысли про документацию

- **Документация должна быть всегда в актуальном состоянии!**
 - лучше написать кратко, но регулярно обновлять описание, чем написать подробно и не обновлять
- **Культура ведения документации в команде**
 - все участники процесса могут вести документацию – это удобно
- **Техническая документация – важна!**
 - комментарии в коде не всегда дают общее понимание процесса
- **Больше схем и диаграмм**
 - визуальная информация легче воспринимается



Общие мысли про документацию

Документация должна быть написана так, чтобы можно было передать проект другой команде без личного общения!

