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

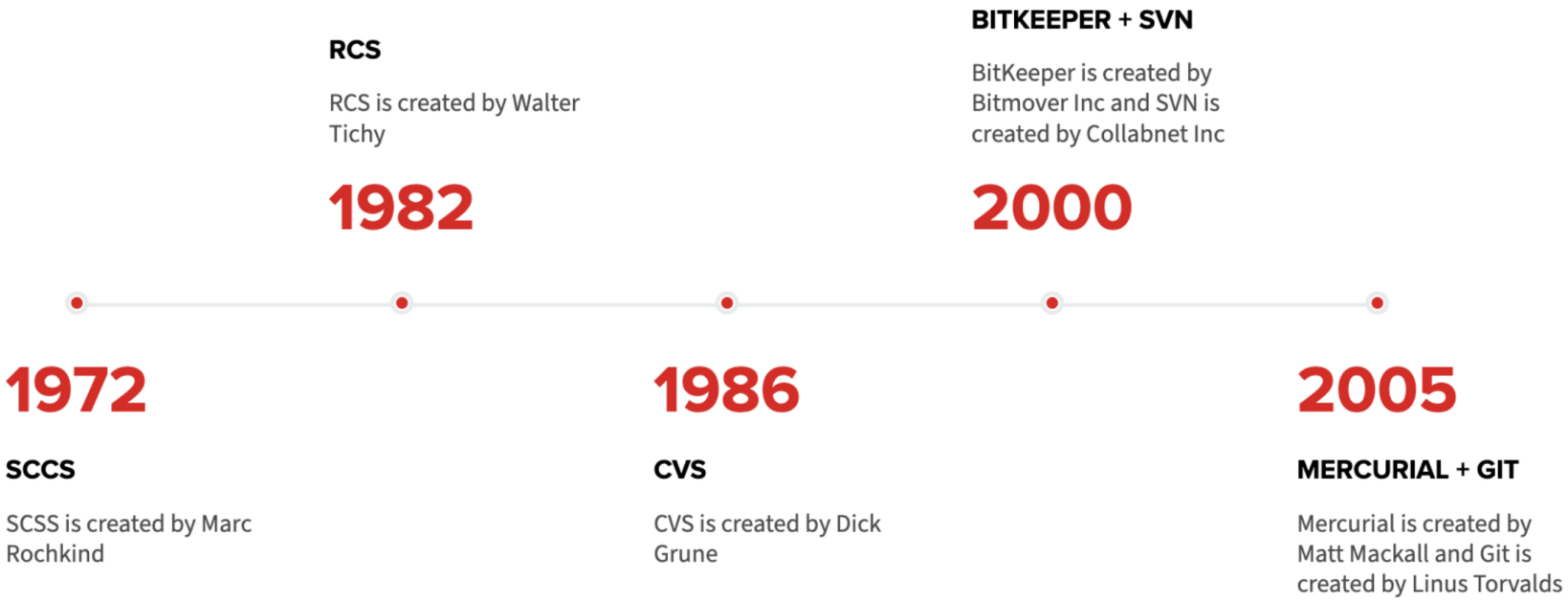
Системы контроля версий

- 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 чужие наработки, следить за новостями других пользователей

GitHub Desktop

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

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

На курсе будем использовать GitHub Desktop.

Начало работы

1. Регистрируемся на сайте github.com
2. Устанавливаем программу GitHub Desktop 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_practical_ML

- Клонируем себе репозиторий
- Учимся проверять, есть ли обновления

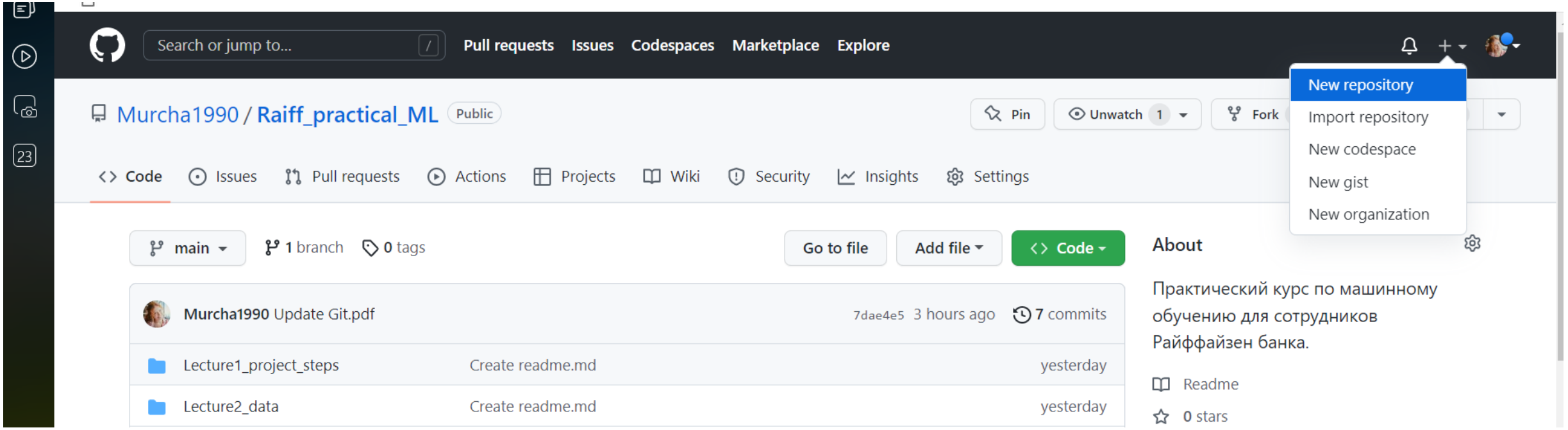
Репозиторий с материалами курса

- Изучаем репозиторий

https://github.com/Murcha1990/Raiff_practical_ML

- Клонируем себе репозиторий
- Учимся проверять, есть ли обновления

Как создать репозиторий?



При создании репозитория обязательно ставьте галочку “create README”!

Домашнее задание (первая часть)

Шаг 1

- Создаем репозиторий командного проекта, называем осмысленно
- Заполняем README описанием задачи (копируем туда описание данных и самой задачи)
- Клонируем 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

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**

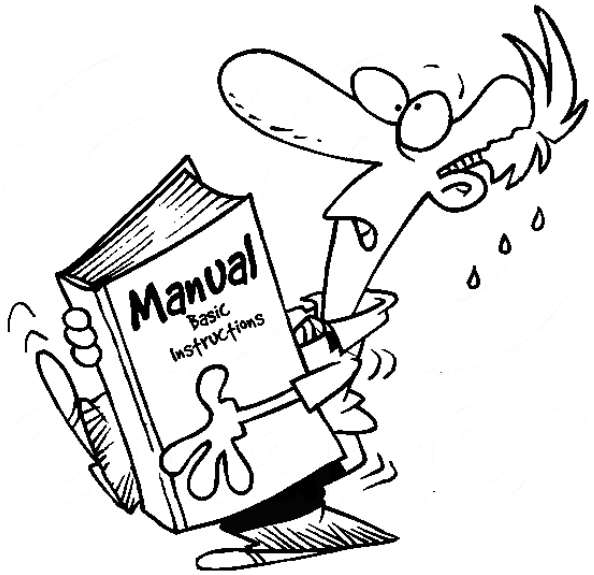
Что писать?

- 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,

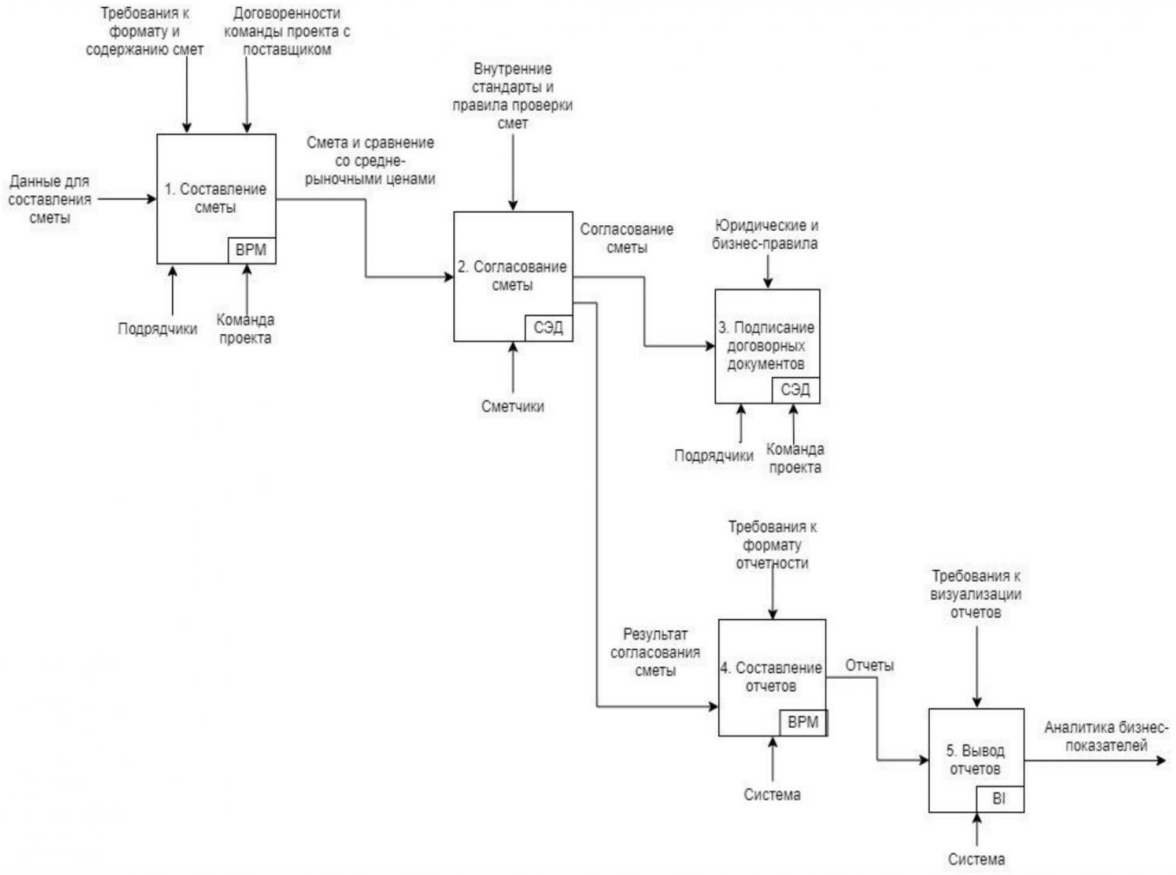
верхнеуровневая модель

функциональности,

взаимодействие с другими системами

6. **Классы пользователейи 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!