

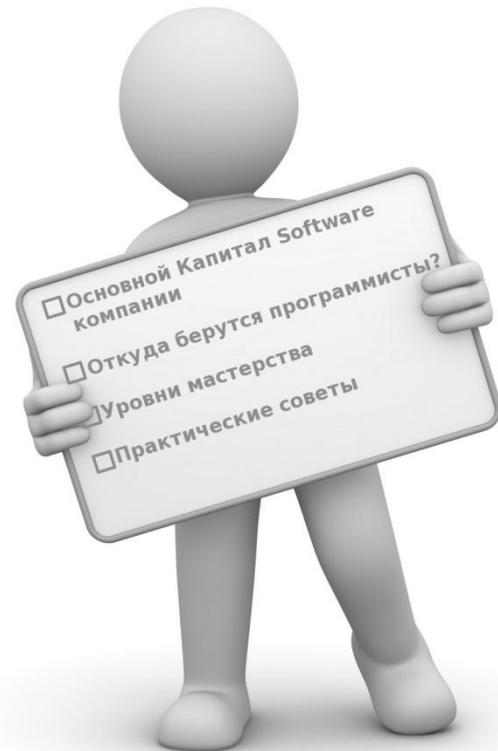
# Как воспитать программиста

Несколько теоретических и три  
практических совета по  
воспитанию эффективного  
программиста

Михаил Пайсон  
mikhail@payson.ru

# План

- Основной капитал Software компании
- Откуда берутся программисты?
  - Как найти готового специалиста?
  - Как найти студента?
  - Как сделать что-то с уже существующими людьми?
- Уровни мастерства
  - Источники плохого кода
  - Три шага становления программиста
- Практические советы
  - Чему учить?
  - Основные ценности
  - Инспекция кода



# Основной капитал Software компании

Что является  
основным продуктом  
Software компании?



# Основной капитал Software КОМПАНИИ

## Программный продукт!

- Производство
- Интеграция
- Поддержка
- Маркетинг и продажи



# Основной капитал Software КОМПАНИИ

- Какие ресурсы идут на производство продукта?



# Основной капитал Software компании

Единственное, что  
требуется для  
производства ПО -

**Человеческая  
мысль!**



# Основной капитал Software КОМПАНИИ

Основной капитал  
Software компании-

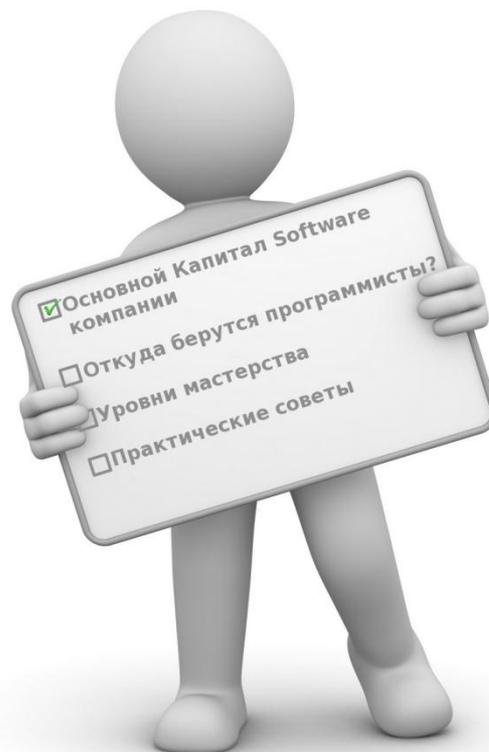
**люди, которые в  
ней работают**

А конкретно –  
программисты,  
которые пишут ПО



# Откуда берутся программисты?

- Как найти готового специалиста?
- Как найти студента?
- Как сделать что-то с уже существующими людьми?



# Откуда берутся программисты?

Как найти готового специалиста?



# Откуда берутся программисты?

## Как найти готового специалиста?

- Переманить деньгами
- Переманить интересной работой
- Переманить карьерным ростом
- Переманить условиями труда
- Получить готового специалиста в связи с его переездом



# Откуда берутся программисты?

Как найти студента?



# Откуда берутся программисты?

## Как найти студента?

- Создать высокий имидж компании
- Договориться с кафедрами
- Прочитать курс лекций
- Организовать СКБ или НИРС



# Откуда берутся программисты?

Как сделать что-то с уже существующими людьми?



# Откуда берутся программисты?

Как сделать что-то с уже существующими людьми?

**Учить**

**Направлять**

**Воодушевлять**

**Контролировать!**



# Откуда берутся программисты?

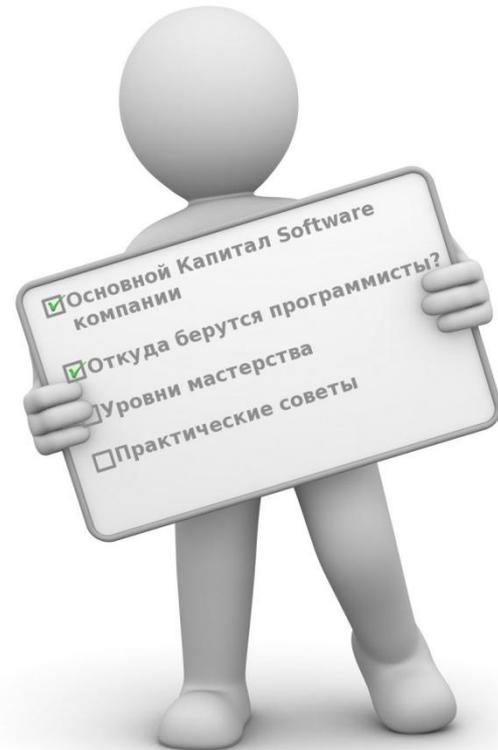
## Кейс

- Вы приходите в новую группу
- Есть разработчик:
  - Большой опыт и знания
  - Не удовлетворяет начальство, т.к. не может в срок выполнить задачу
  - «Слил» важный проект
- Решение начальства: «надо увольнять»
- Ваши действия?



# Уровни мастерства

- Источники плохого кода
- Три шага становления программиста
- Кривая обучения

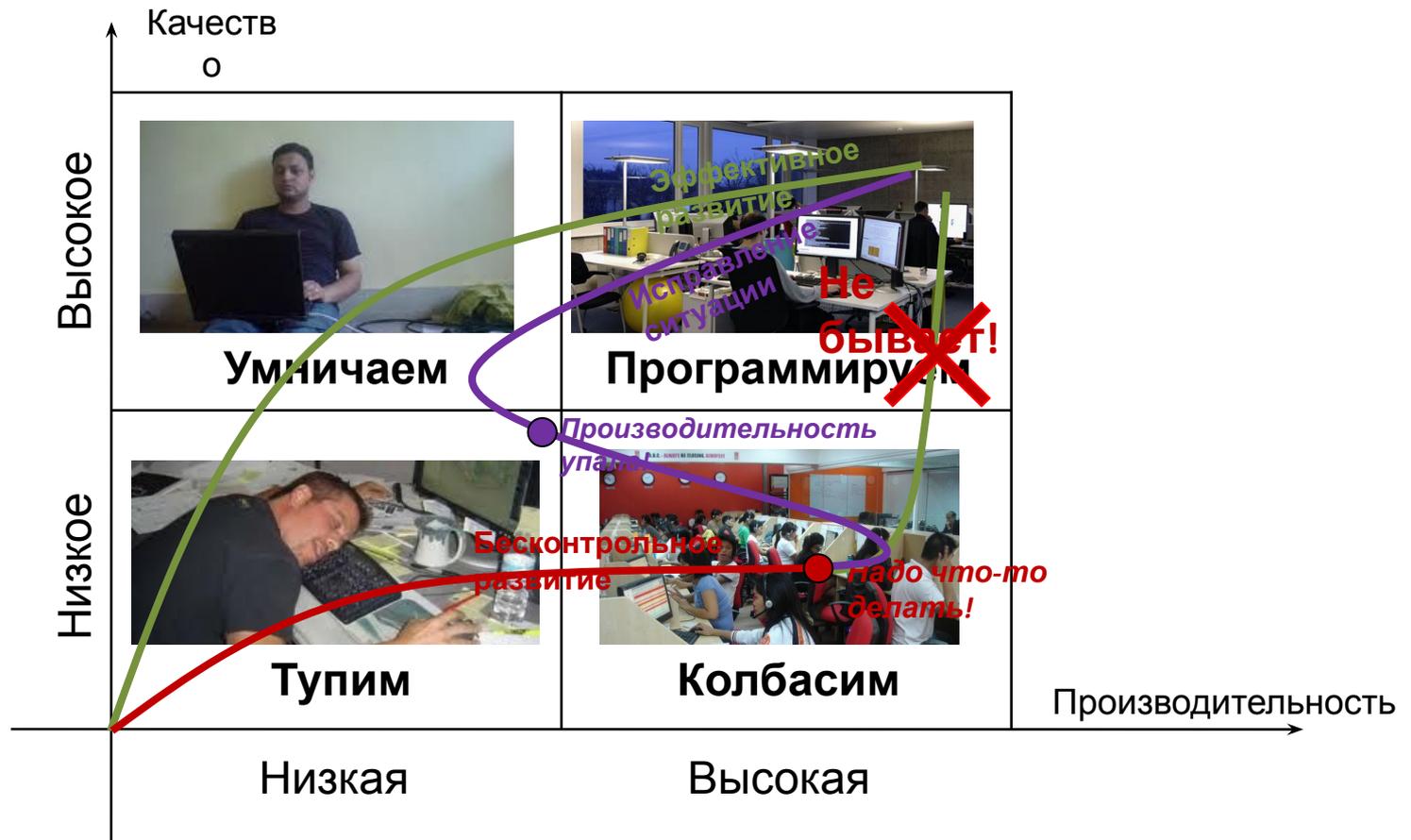


# Источники плохого кода

- Никто и никогда сознательно не пишет плохо
- Основной источник – отсутствие «воспитания»
- Второй источник – отсутствие контроля
- Третий источник – отсутствие времени



# Три шага становления программиста



# Шаг 1. Ненависть

- Научите программиста ненавидеть:
  - Некачественный код
  - «Сделать как побыстрее»
  - Костыли и заплатки
  - «Сейчас начнём, а потом увидим – что получится»



# Шаг 2. Страсть

- Программист начинает:
  - Любить «умничать»
  - Писать «красивые решения»
  - Наворачивать паттерны и методологии
  - Отшлифовывать код до блеска
  - Гордиться своей профессиональной компетентностью



© www.123rf.com

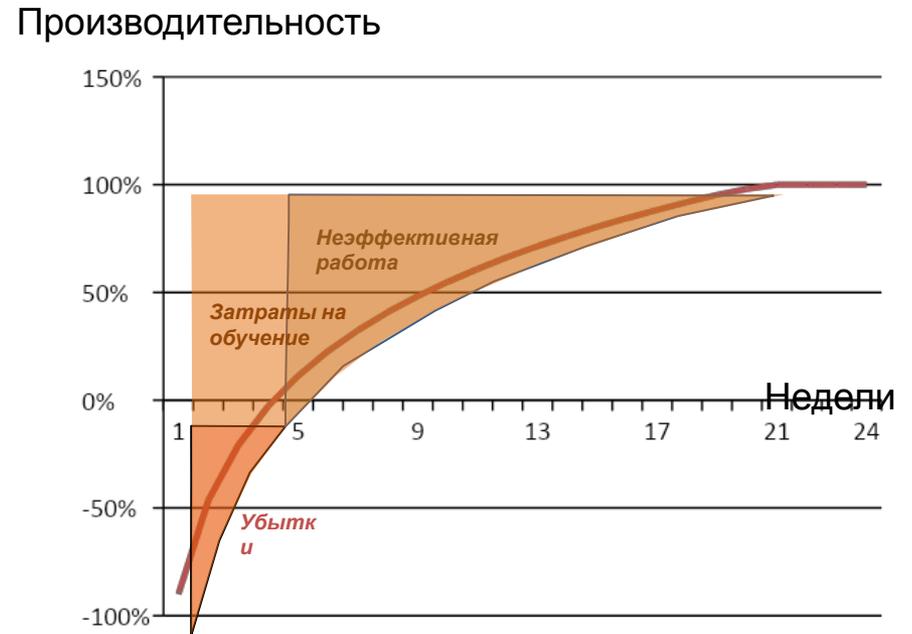
# Шаг 3. Здравомыслие

- Программист осознаёт:
  - Думаем прежде, чем пишем
  - Нет универсальных решений
  - Чем проще, тем легче работать
  - Эффективность прежде всего



# Кривая обучения

- Первое время любой программист тратит время команды
- С определённого момента он начинает работать сам, но работает неэффективно
- Для того, чтобы программист вышел на свою полную мощность требуется много времени
- **Опять инвестиции!**



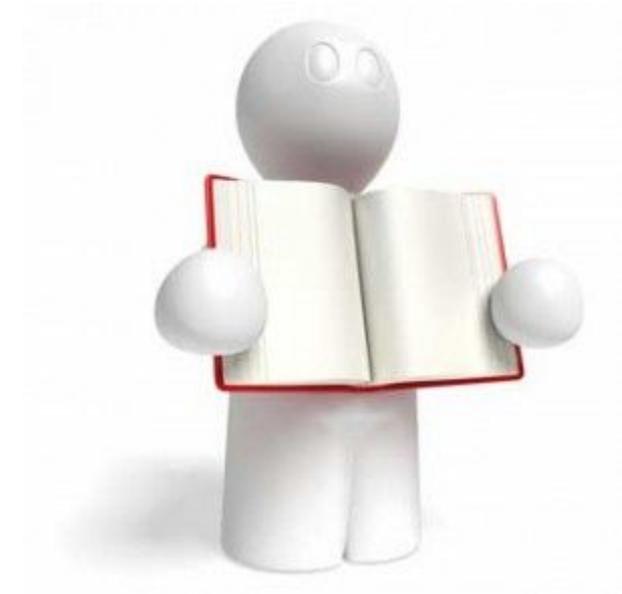
# Практические советы

- Чему учить?
- Основные ценности
- Инспекция кода



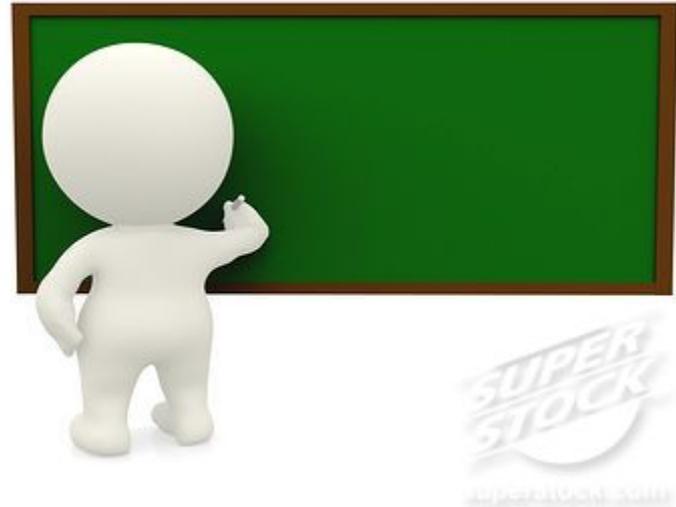
# Чему учить?

- Подумай каждый раз перед тем, как писать
- Код вероятно придётся модифицировать: чем «грязнее» код, тем больше проблем у тебя будет потом
- Аккуратное приложение без ошибок выглядит круто. И все любят его автора.
- Трус не играет в хоккей, а профессионал не «колбасит»



# Чему учить?

- Общие соглашения написания кода (унификация)
- Низкоуровневая архитектура (паттерны)
- Использование сторонних компонентов и готовых решений



# Основные ценности

Код должен быть

- Продуман
- Аккуратен
- Эффективен
  
- **Быстро, просто и аккуратно решать поставленную задачу**



# Инспекция кода

- Проводится еженедельно для кода, написанного за неделю
- Проводится опытными программистами по очереди
- Время на подготовку – 4 часа
- Время на проведение – 1 час



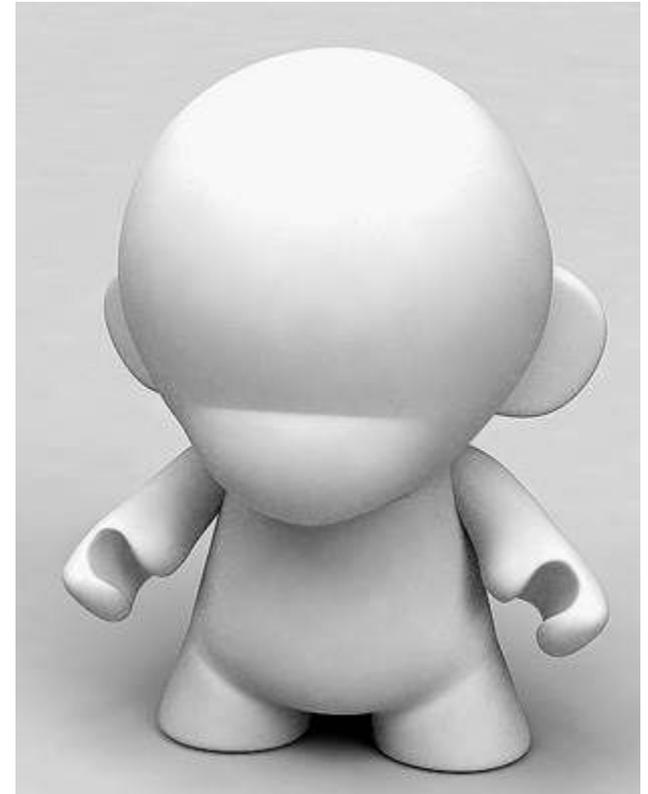
# Инспекция кода. Проведение

- Проводится в виде обсуждения
- Ведущий готовит список недочётов (по его мнению)
- Имена авторов «плохого кода» не указываются
- Команда обсуждает каждый из недочётов
- Обязательно подводятся итоги
- Принимается решение о исправлении либо рефакторинге части недочётов



# Инспекция кода. Junior

- Для junior
  - Нахождение «плохого кода», пока он не врос в систему
  - контроль профессионального развития программиста
  - Обучение: обсуждение и применение подходящей архитектуры



# Инспекция кода. Senior

- Для senior
  - «Свежий взгляд» на код
  - Обдумывание более быстрых и эффективных решений
  - Иногда и профессионалы «колбасят» 😞



# Инспекция кода. Команда

- Для команды в целом
  - Возможность обсудить качество кода и архитектурных решений
  - Стимул писать аккуратно, чтобы потом не краснеть на review
  - Постоянное поддержание необходимости «писать хорошо»



# Инспекция кода. Проблемы

- Поиск серьёзных архитектурных недочётов малоэффективен
- Очень сложно избежать перехода на личности
- Вечный недостаток времени



# Обсуждение

