Технологии проектирования ИС и ИТ

ФИО преподавателя: Смирнов Михаил Вячеславович

e-mail: smirnov.mirea@gmail.com

Лекция 4

Гибкая разработка программного обеспечения

Содержание

- “Темный манифест Agile” и ковбойский кодинг
- Типовой проект разработки ПО с точки зрения аккуратного классика, ковбоя и “эджайл энджоера”
- Модель SCRUM-фреймворка
- Участники SCRUM
- Функциональные требования SCRUM и пользовательские истории (PBI)
- Управление и масштабирование backlog SCRUM

Темный манифест Agile

Dark Manifesto for Agile Software Development

We are uncovering ~~better~~ **the only** ways of developing
software by ~~doing it and helping~~ **teaching** others ~~do it~~.
Through this work we have come to value:

Individuals and interactions ~~over~~ **and not** processes and tools
Working software ~~over~~ **and not** comprehensive documentation
Customer collaboration ~~over~~ **and not** contract negotiation
Responding to change ~~over~~ **and not** following a plan

That is, ~~while~~ **since** there is **no** value in the items
on the right, we value **only** the items on the left ~~more~~.

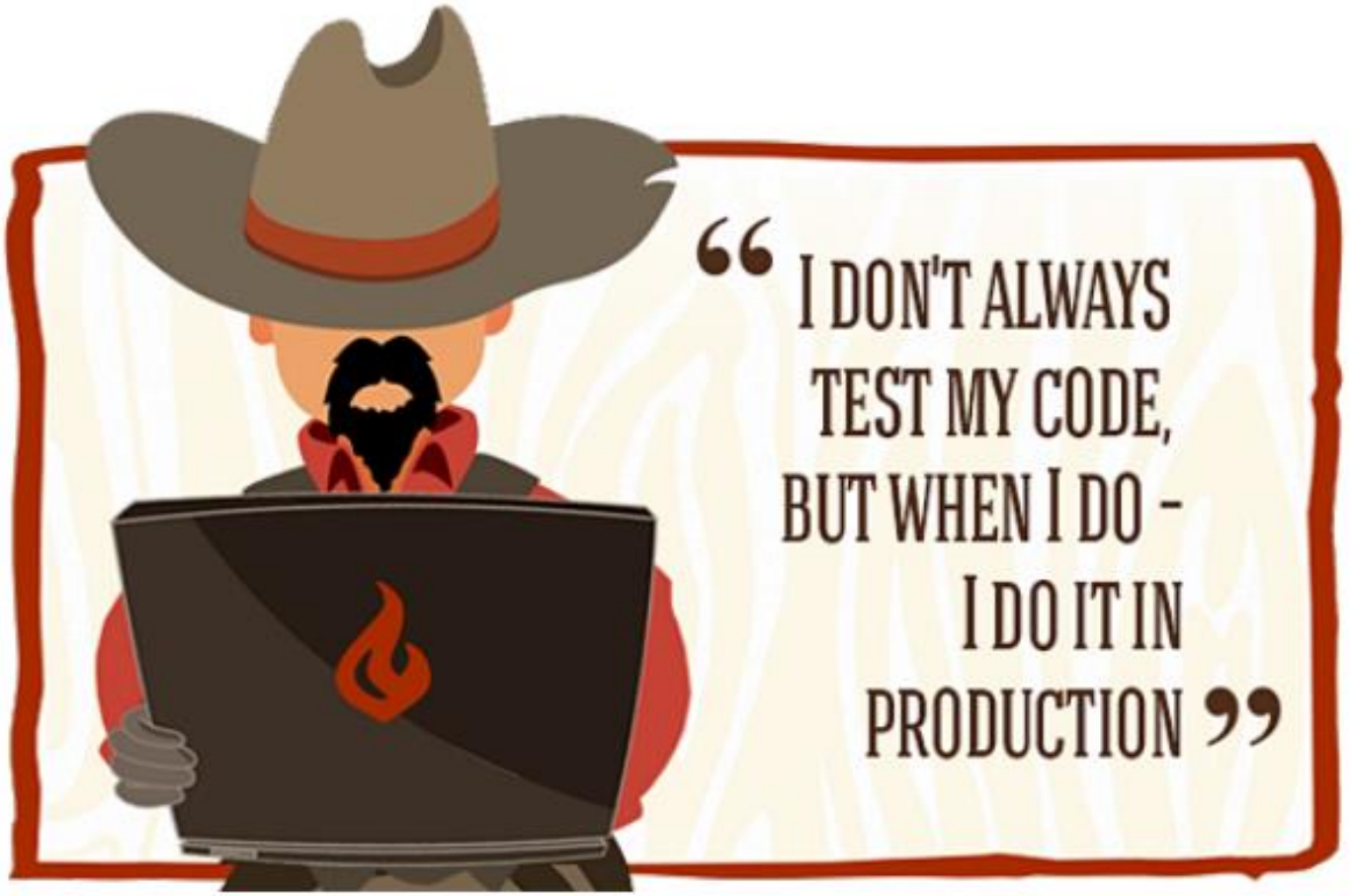
The paper and the slides we presented at SPLASH 2012
Participate in our questionnaire about the Dark Agile Manifesto

3

Темный манифест Agile

- Такая “интерпретация”… … защищает программирование без дисциплины, планирования и документирования: “это не что иное, как попытка узаконить хакерское поведение”
- Сообщество Agile разработало еще один термин, чтобы описать такое поведение и отделить себя от него:

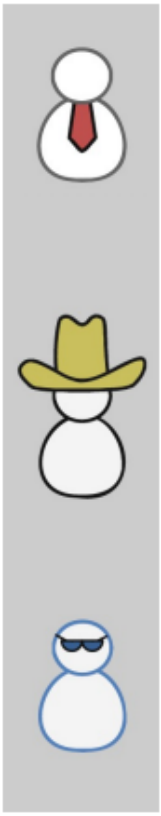
Ковбойский кодинг



Планирование проекта

- Планировать все хорошо и заранее чтобы потом ничего не менять! Изменения – признак плохого планирования.
- Не надо планировать (менять тоже не надо) Изменения – признак докучающего менеджера
- Планировать то, что можно предвидеть в разумных пределах и быть готовым к изменениям!

Анализ и дизайн продукта



- Нужен полный анализ и дизайн перед началом создания продукта.
- Не нужен анализ и дизайн, где компьютер? Начинаем программировать
- Не стоит делать полный анализ и дизайн для того чтобы начать программировать.

Общение с заказчиком

- Заказчик должен видеть только готовый продукт. Сырой продукт признак не профессионализма!
- Стараться показать версию продукта как можно позднее. Не надо терять времени на разговоры!
- Как можно чаще показывать заказчику разработку, иметь представителя от заказчика в команде разработки на постоянной основе.

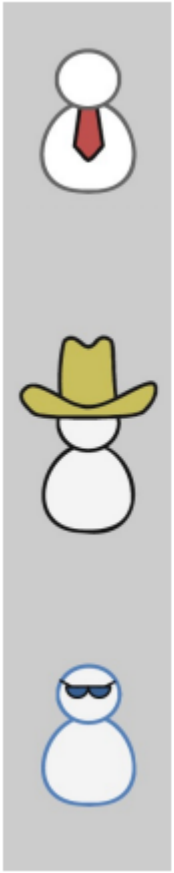
Изменения требований

– Изменения зло! Надо бороться с ними!

– Изменения зло! Надо бороться с ними!

– Изменения неизбежны, нужно справляться с ними

Отношение к коду

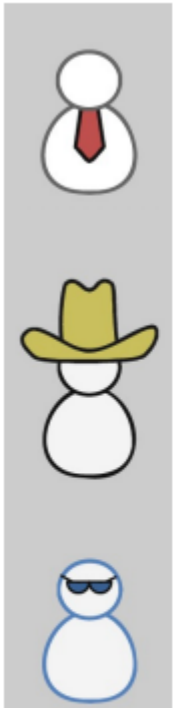


– Если не сломано, не трогай!

– Даже если сломано, не трогай! Надо спрятать!

– Даже если не сломано, постоянно улучшай!
Не забывай о тестировании изменений.

Срок сдачи продукта



- К концу срока работать как сумасшедшие, чтобы завершить проект. Программировать тяжело.
- К концу срока работать как сумасшедшие, чтобы завершить проект. Программировать весело.
- Сдавать продукт часто. Работать не более 40 часов в неделю, сохранять постоянную скорость и свежий ум.

Интеграция продукта

– Интеграция после завершения разработки! Будет тяжело, нужно подготовить полную детализированную документацию и протоколы

– После завершения разработки будем интегрировать Никаких проблем, легко: займет 5 минут!

– Производить интеграцию ежедневно, тогда в конце разработки не возникнет авралов и проблем.

Модель процессов фреймворка SCRUM

Работа SCRUM фреймворка

Картинка кликабельна ;)

Владелец продукта

- Единолично несет ответственность за принятие решений:
  - какой функционал разрабатывать
  - в каком порядке его разрабатывать
- Обеспечивает четкое виденье продукта.
- Отвечает за общий успех разрабатываемого или поддерживаемого решения.

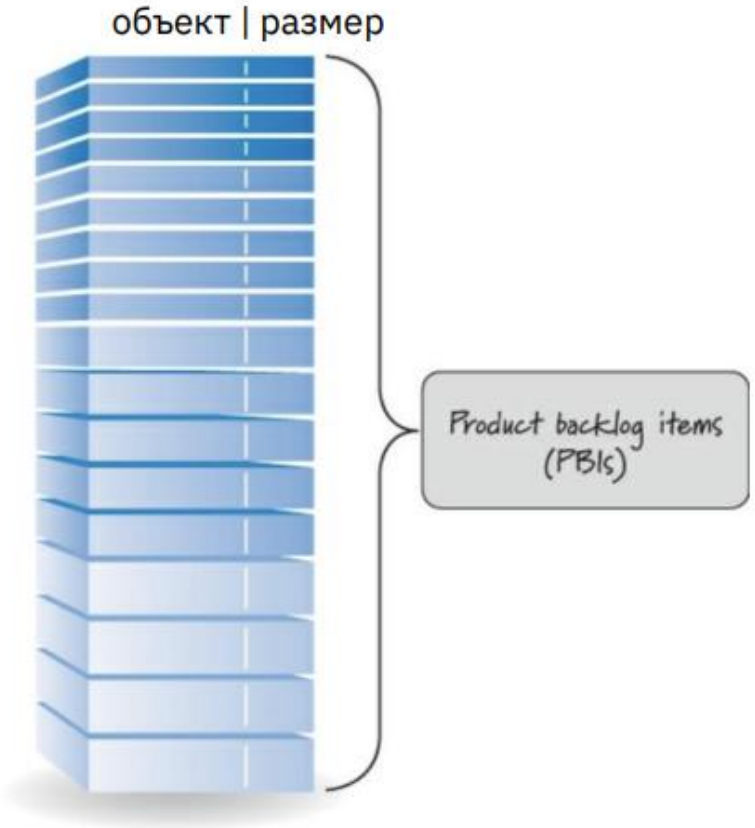
SCRUM мастер

- Помогает понять и использовать по максимуму ценности, принципы и практики SCRUM.
- Выступает в качестве тренера, лидера (но не менеджера)
  - помогает организовать процесс управления изменениями
  - помогает решать возникающие проблемы
  - защищает команду от внешнего вмешательства
- Не имеет полномочий контроля над командой.

Команда разработки

- Состоящая из 5-9 человек, является самоорганизованной и кросс-функциональной.
- Разработчики объединяют роли архитектора, программиста, тестировщика, администратора баз данных, дизайнера пользовательских интерфейсов и т.д..
- Отвечают за проектирование, создание и тестирование программного продукта.

Backlog SCRUM (журнал функциональных требований)



– Функциональное требование
– Изменение
– Дефект
– Техническое улучшение
– Приобретение знаний

Функциональные требования в SCRUM оформляются как Пользовательские истории (user stories)

Структура типовой пользовательской истории

- Как… (кто заинтересован в функционале)
- Я хочу… (что представляет функционал)
- Для того,… (почему это важно сделать)

Хорошая пользовательская история

Как начальник, я хочу установить умные датчики на узлы переработки, для того чтобы компания использовала умные датчики

Как инженер, я хочу автоматический сбор показаний на узлах переработки, для того чтобы исключить человеческий фактор

PBI: Изменение

Изменение: как представитель службы поддержки клиентов, я хочу, чтобы по умолчанию результаты поиска были по фамилии, а не по номерам заявок, чтобы было легче найти заявку адресованную к техподдержке.

PBI: Дефект

Дефект: исправить дефект #UF256 в системе отслеживания дефектов, для того, чтобы спец-символы в поисковых запросах пользователей не приводили к системному сбою.

PBI: Техническое улучшение

Техническое улучшение: как разработчик, я хочу произвести миграцию системы для поддержки работы с последней версией базы данных Oracle, таким образом, мы избежим последствий работы с версией БД, которая в ближайшем времени будет снята с поддержки.

PBI: Приобретение знаний

Приобретение знаний: как архитектор, я хочу создать прототип системы для проверки двух архитектурных решений, а также запустить нагрузочное тестирование, для того чтобы понять, какое решение будет справляться лучше.

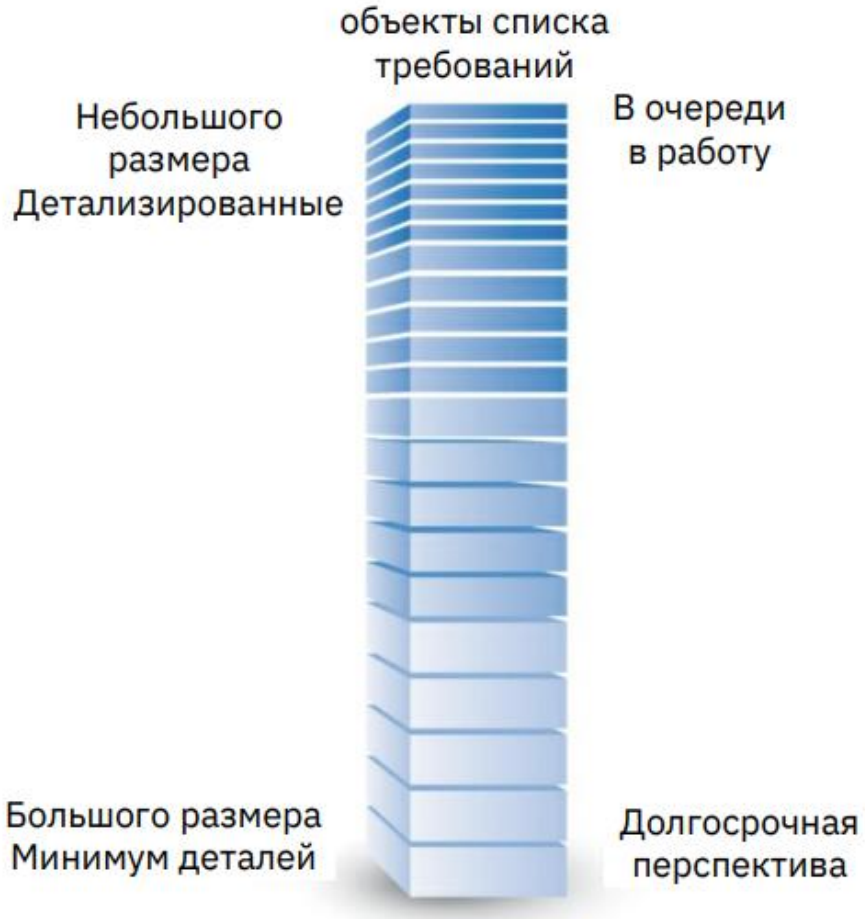
Характеристики хорошего списка требований

Роман Пихлер и Майк Кон придумали акроним DEEP (глубокий):

- Detailed appropriately – Правильно детализированный
- Emergent – Постоянно развивающийся
- Estimated – Предварительно оцененный
- Prioritized – Приоритезированный

Правильная детализация

Не все объекты в списке требований находятся одновременно на одном уровне детализации



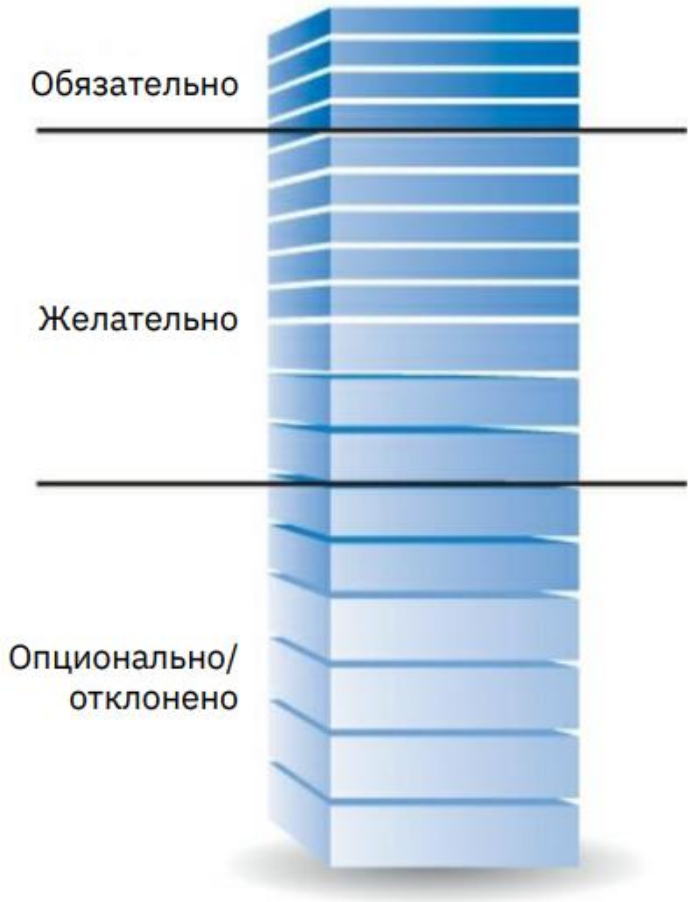
Расстановка приоритетов

Обязательно – критично для разработки

Желательно – важно, но не критично для разработки

Опционально – наименее критичные объекты

Отклонено – объекты которые не будут разрабатываться



Предварительно оцененный

Владелец продукта использует предварительную оценку для расстановки приоритетов и планирования

Практика оценки

Расставить приоритет в последовательности:

1. Эпики
2. Фичи в эпиках
3. Функциональные требования
4. Объекты в списке спринта

Постоянное развитие

Список требований должен быть всегда актуальным.

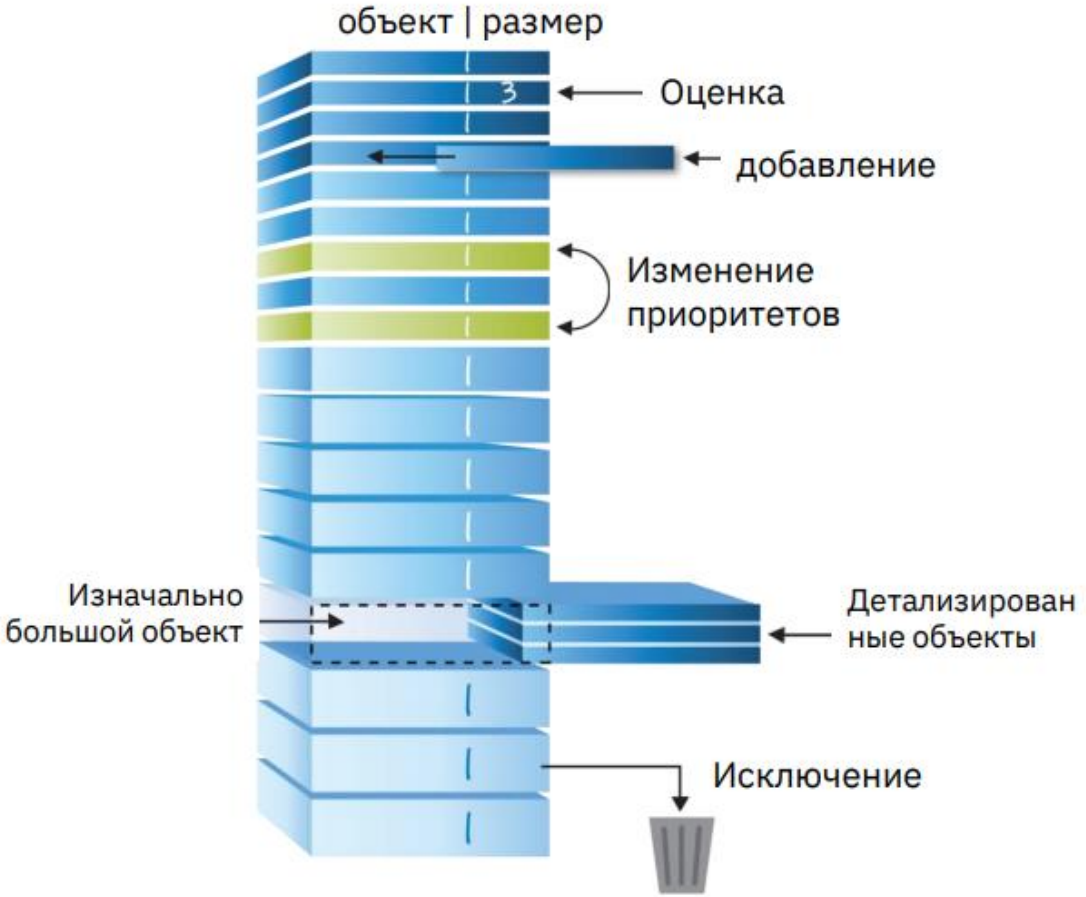
Необходимо планово проводить:

– Переоценку объектов

– Обновление приоритетов

– Добавление новых объектов

– Детализацию объектов подходящих к выполнению



Единицы измерения пользовательских историй

Story Points (стори поинты) – единица измерения, которая комбинирует фактор сложности и размер элемента списка требований. Стори поинты комбинируют фактор сложности и размер элемента списка требования. Позволяют абстрагироваться от профессионализма разработчика

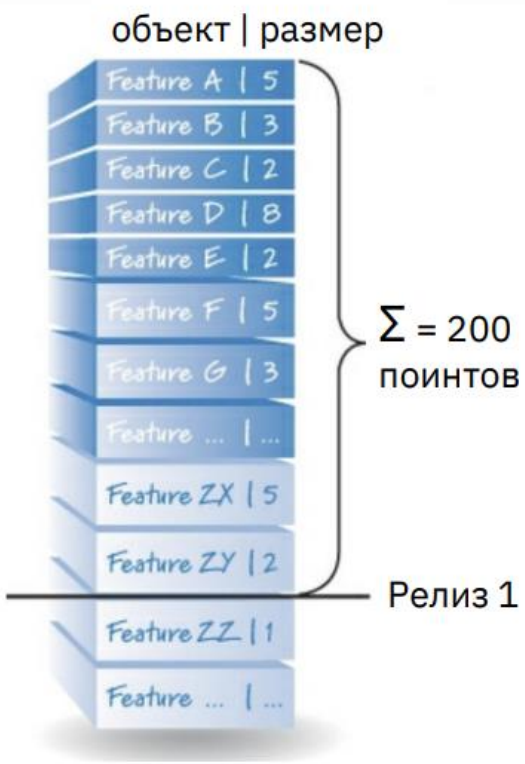
Большой ≠ сложный и маленький ≠ легкий

Ideal Days (человеко-часы) – отображают количество часов необходимых для выполнения задачи. Нет абстрагирования от профессионализма разработчика.

Человеко - часы ≠ затраченное время

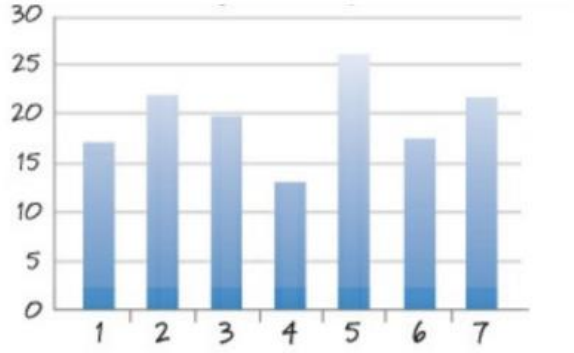
Размер проекта и результативность команды

Спрогнозированный размер ÷ измеренная продуктивность команды = количество итераций



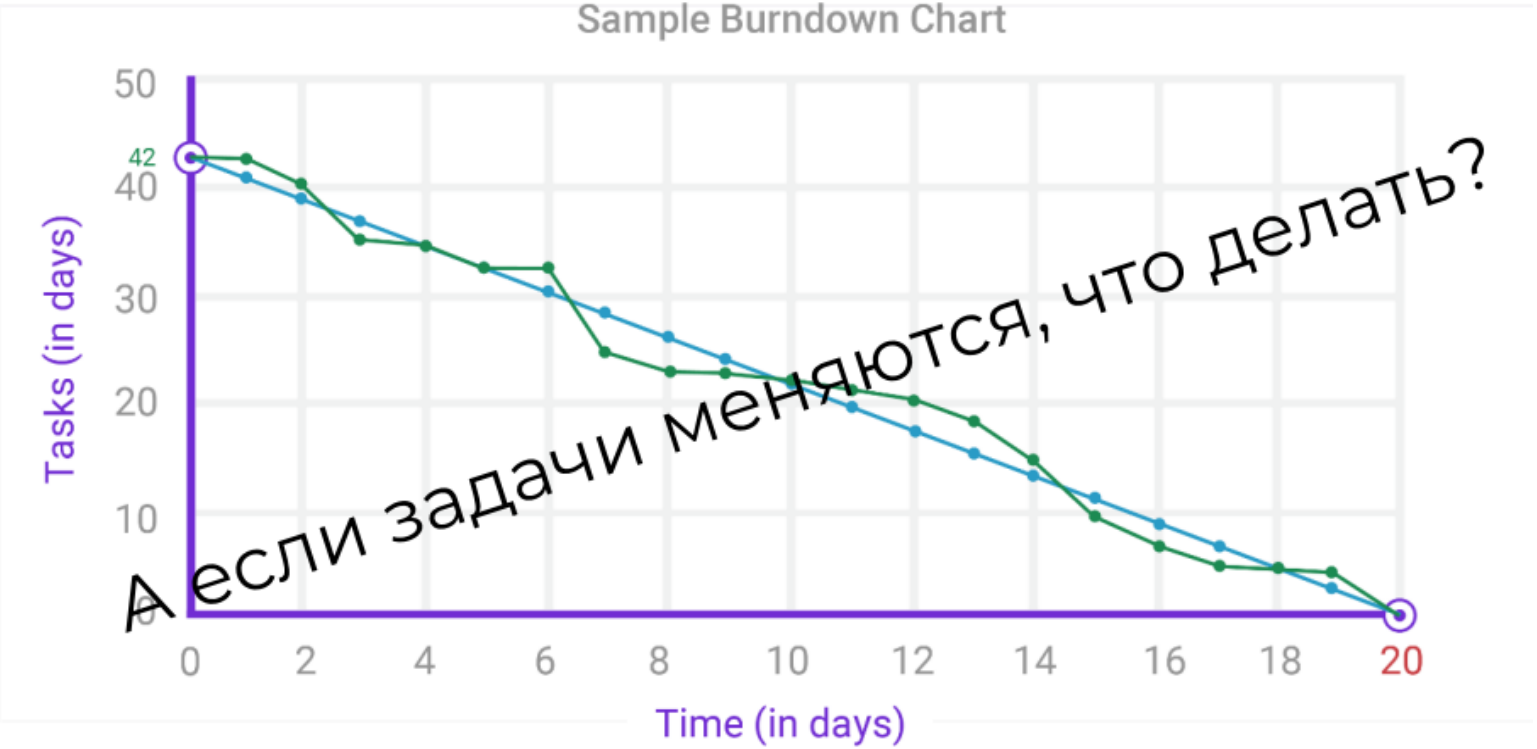
200 поинтов ÷ 20 поинтов/итерация= 10 итераций

Средняя продуктивность* = 20

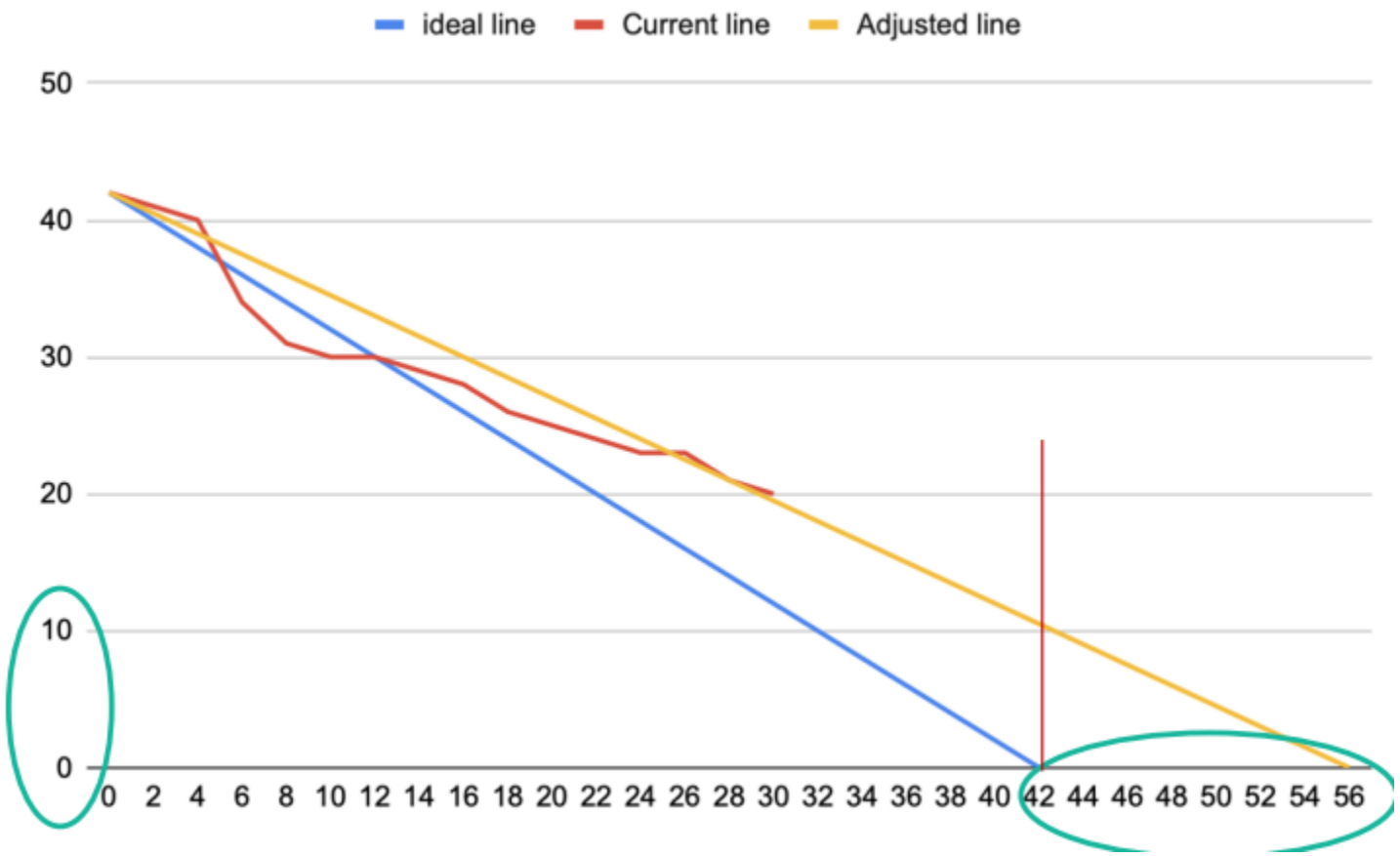


*team velocity – разгон команды

Burndown chart (ключевой элемент мониторинга гибкого проекта)



Корректировка burndown chart



Спасибо за внимание!