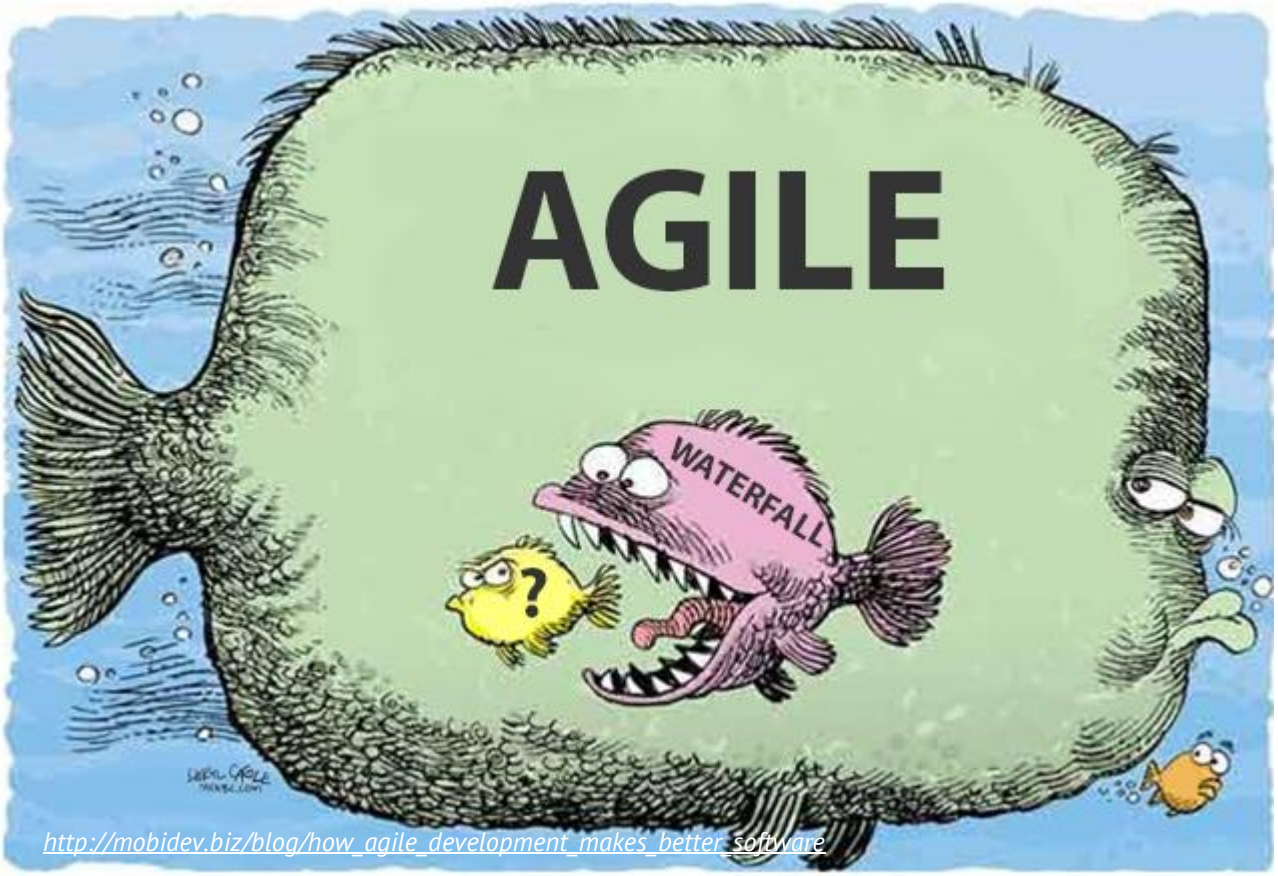


http://mobidev.biz/blog/how_agile_development_makes_better_software

Курс: «Математические модели комплексов программ»

Модуль: **«Архитектура программных систем»**

Лекция 2 (Князьков К.В.)

Гибкая методология разработки

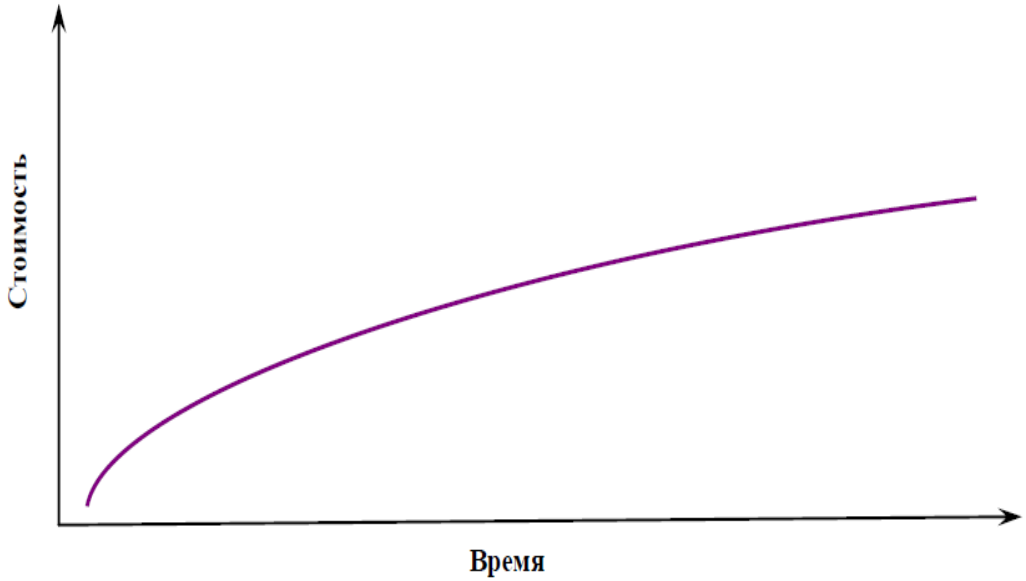
Гибкая методология разработки (англ. Agile software development, agile-методы) — серия подходов к разработке программного обеспечения, ориентированных на использование итеративной разработки, динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля

Примеры:

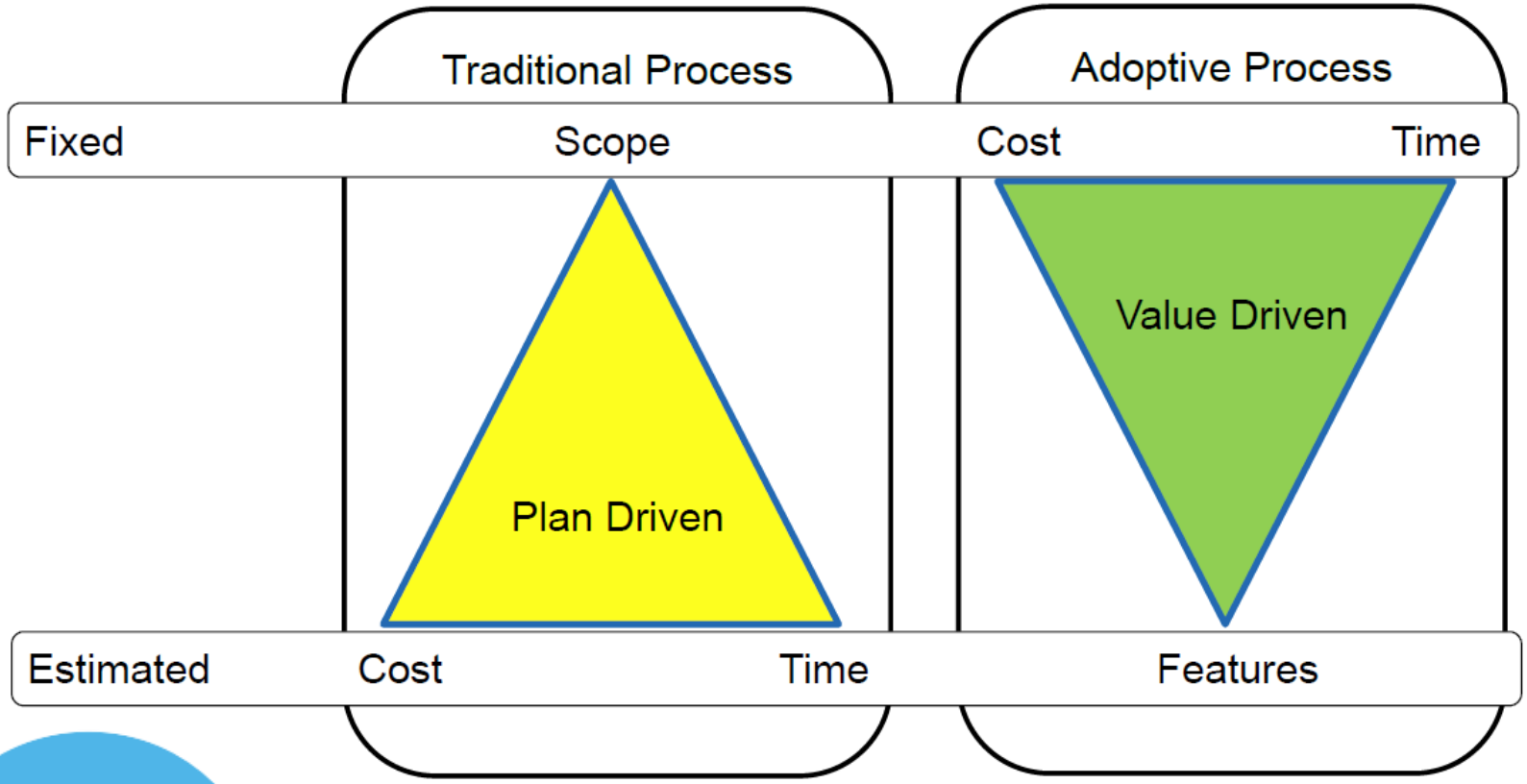
- Экстремальное программирование (XP)
- Scrum
- Feature Driven Development
- Dynamic Systems Development Method
- …

Для чего?

Классическая кривая стоимости изменений

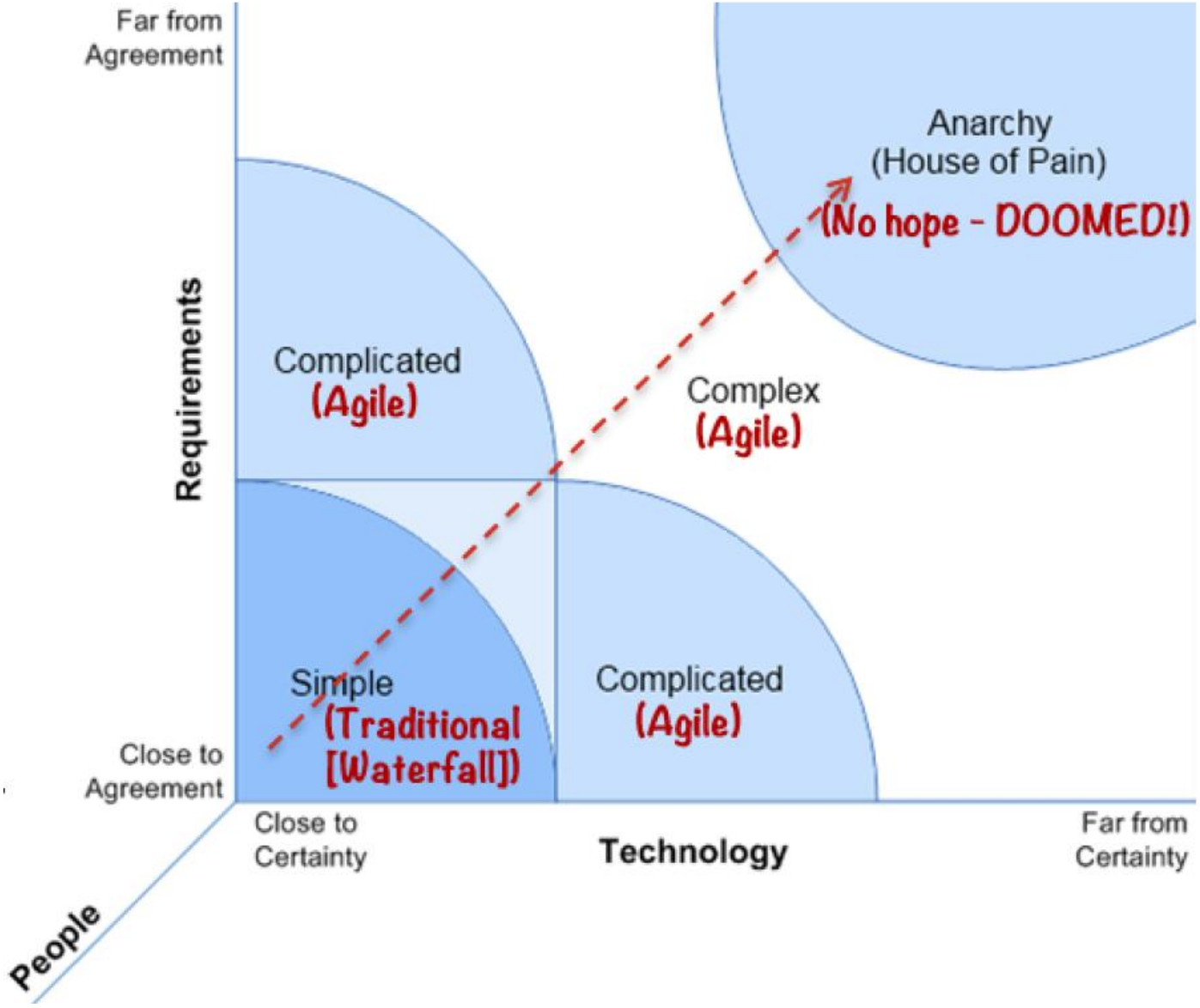


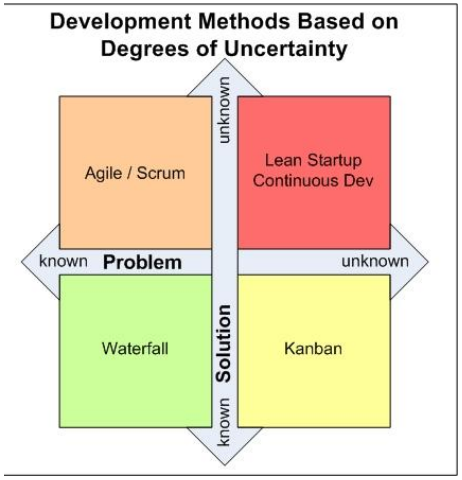
Кривая стоимости изменений в гибких технологиях



http://www.scgmis.org/Resources/Documents/Agile%20Overview%20rev%209-synopsis.pdf

Stacey diagram





http://www.scgmis.org/Resources/Documents/Agile%20Overview%20rev%209-synopsis.pdf

←Response to unpredictability→

- Decide Early
- Deliver Slow

Predictive

Waterfall

Plan-driven

Incremental

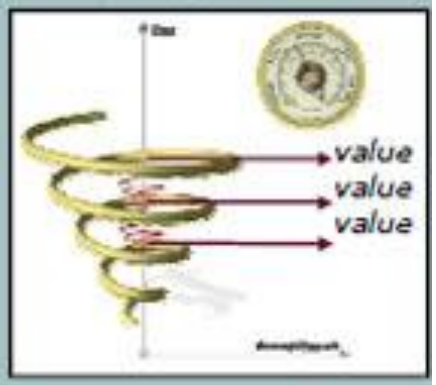
RUP spiral model

Minimizing risk

- Decide Late
- Deliver Fast

Adaptive

Agile



Change-driven

Change Tolerance

http://agilescrum.foundationtraining.nl/scrum-agile-training-agile-essenties.php

Философия Agile

Agile-манифест разработки программного обеспечения

Мы постоянно открываем для себя более совершенные методы разработки программного обеспечения, занимаясь разработкой непосредственно и помогая в этом другим. Благодаря проделанной работе мы смогли осознать, что:

Люди и взаимодействие	важнее	процессов и инструментов
Работающий продукт	важнее	исчерпывающей документации
Сотрудничество с заказчиком	важнее	согласования условий контракта
Готовность к изменениям	важнее	следования первоначальному плану

То есть, не отрицая важности того, что справа,
мы всё-таки больше ценим то, что слева.

http://agilemanifesto.org/iso/ru/

Принципы, которые разъясняет Agile Manifesto

http://www.agilemanifesto.org/principles.html

1. удовлетворение клиента за счёт **ранней** и бесперебойной **поставки** ценного программного обеспечения;
2. приветствие **изменений требований** даже в конце разработки (это может повысить конкурентоспособность полученного продукта);
3. **частая поставка** рабочего программного обеспечения (каждый месяц или неделю или ещё чаще);
4. тесное, ежедневное **общение заказчика** с разработчиками на протяжении всего проекта;
5. проектом занимаются мотивированные личности, которые обеспечены нужными условиями работы, поддержкой и доверием;
6. рекомендуемый метод передачи информации — **личный разговор** (лицом к лицу);

Принципы, которые разъясняет Agile Manifesto

7. работающее программное обеспечение — лучший **измеритель прогресса**;
8. спонсоры, разработчики и пользователи должны иметь возможность поддерживать постоянный **темп** на **неопределённый срок**;
9. постоянное внимание **улучшению технического мастерства** и удобному дизайну;
10. **простота** — искусство не делать лишней работы;
11. лучшие технические требования, дизайн и архитектура получаются у **самоорганизованной команды**;
12. постоянная **адаптация** к изменяющимся обстоятельствам.

http://www.versionone.com/pdf/2013-state-of-agile-survey.pdf

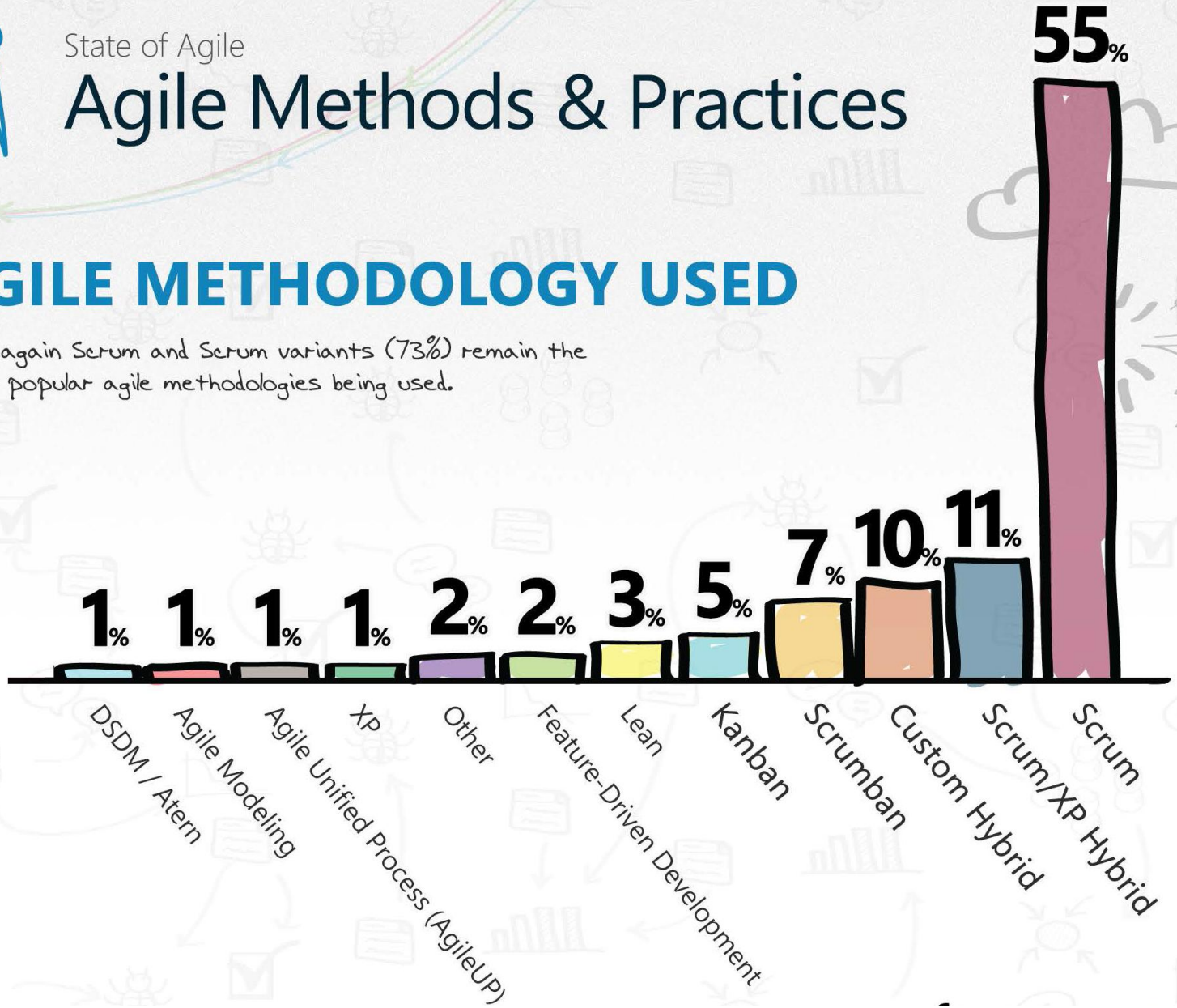
State of Agile

Agile Methods & Practices

8th ANNUAL STATE OF AGILE™ SURVEY

AGILE METHODOLOGY USED

Once again Scrum and Scrum variants (73%) remain the most popular agile methodologies being used.



12 практик Экстремального программирования

- Короткий цикл обратной связи (Fine-scale feedback)
  - **Разработка через тестирование (Test-driven development)**
  - Игра в планирование (Planning game)
  - Заказчик всегда рядом (Whole team, Onsite customer)
  - **Парное программирование (Pair programming)**
- Непрерывный, а не пакетный процесс
  - **Непрерывная интеграция (Continuous integration)**
  - **Рефакторинг (Design improvement, Refactoring)**
  - Частые небольшие релизы (Small releases)
- Понимание, разделяемое всеми
  - Простота (Simple design)
  - Метафора системы (System metaphor)
  - Коллективное владение кодом (Collective code ownership) или выбранными шаблонами проектирования (Collective patterns ownership)
  - Стандарт кодирования (Coding standard or Coding conventions)
- Социальная защищенность программиста (Programmer welfare):
  - 40-часовая рабочая неделя (Sustainable pace, Forty-hour week)

Парное программирование

Преимущества

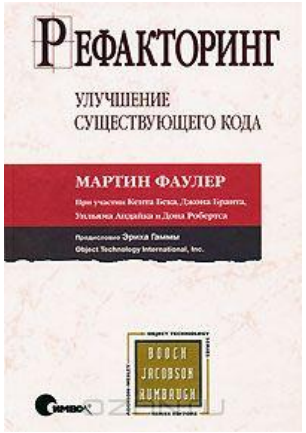
- Экономическая целесообразность (исследования Alistair Cockbern)
- Удовлетворение от работы
- Улучшение качества дизайна
- Повышение дисциплины
- Борьба с отвлекающими факторами
- Улучшение качества кода
- Коллективное владение кодом
- Формирование команды

Исследования эффективности ПП

- by Laurie Williams, 2001
  - время +15%, количество ошибок -15%
- by Arisholm, 2007
  - сложных систем: -48% ошибок, ~0% разница во времени
  - простые системы: -20% время, ~0% разница в количестве ошибок
- by John T. Nosek, 2006
  - Выполнение задачи за 45 мин.: читаемость +30%, функциональность +23%

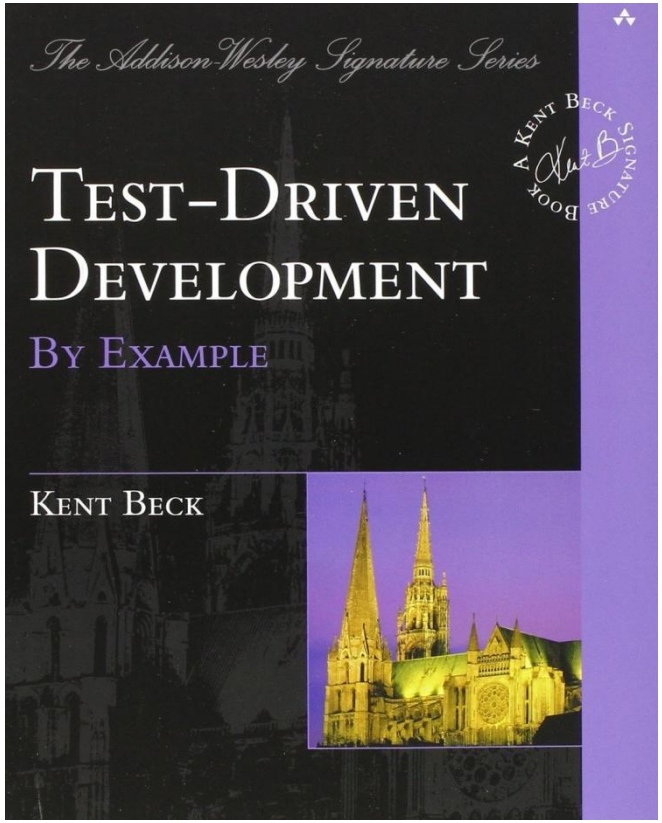
Рефакторинг

Рефа́кторинг (англ. refactoring) или реорганизация кода — процесс изменения внутренней структуры программы, не затрагивающий её внешнего поведения и имеющий целью облегчить понимание её работы.



- Изменение сигнатуры метода (Change Method Signature)
- Инкапсуляция поля (Encapsulate Field)
- Выделение класса (Extract Class)
- Выделение интерфейса (Extract Interface)
- Выделение локальной переменной (Extract Local Variable)
- Выделение метода (Extract Method)
- Генерализация типа (Generalize Type)
- Встраивание (Inline)
- Введение фабрики (Introduce Factory)
- Введение параметра (Introduce Parameter)
- Подъём метода (Pull Up Method)
- Спуск метода (Push Down Method)
- Переименование метода (Rename Method)
- Перемещение метода (Move Method)
- Замена условного оператора полиморфизмом (Replace Conditional with Polymorphism)
- Замена наследования делегированием (Replace Inheritance with Delegation)
- Замена кода типа подклассами (Replace Type Code with Subclasses)

Разработка через тестирование (TDD)

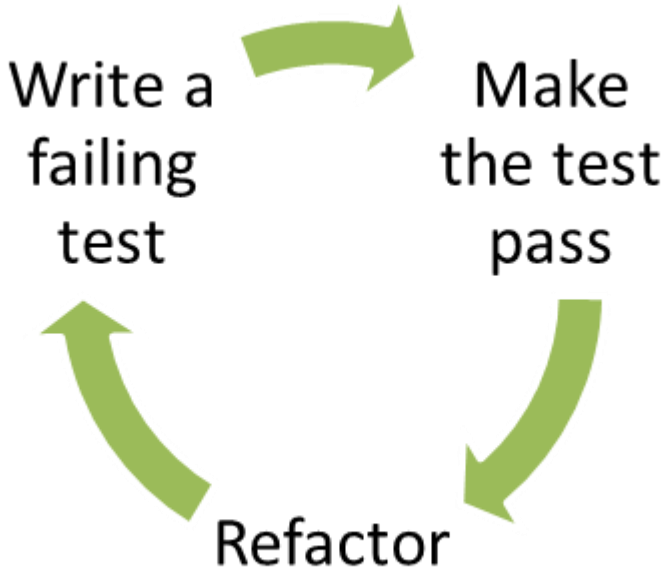


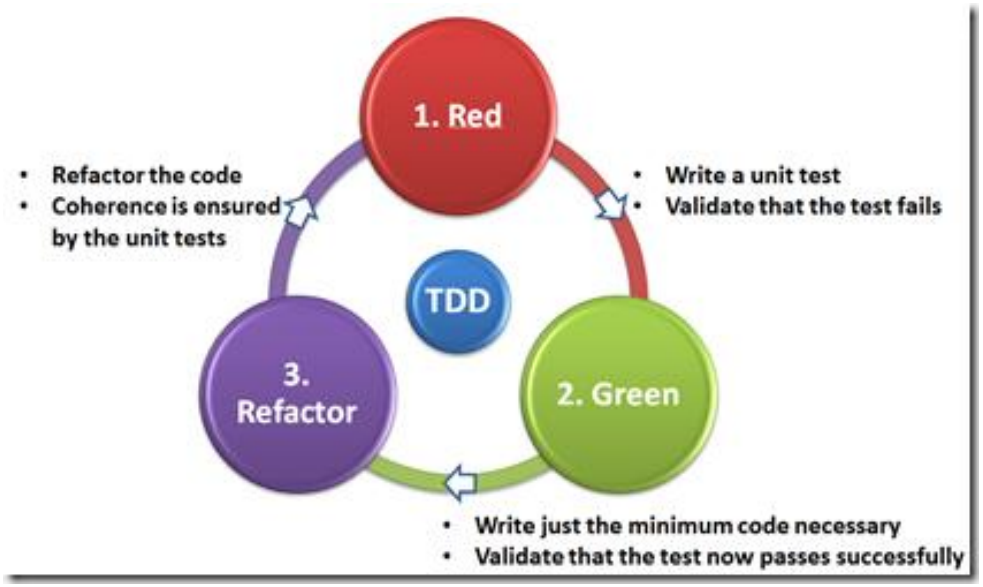
Кент Бек

Kent Beck

Цикл разработки в TDD

1. Написание теста
2. Запуск всех тестов
3. Написание нового кода
4. Запуск всех тестов
5. Рефакторинг
6. *Goto 1*





Непрерывная интеграция (continuous integration)

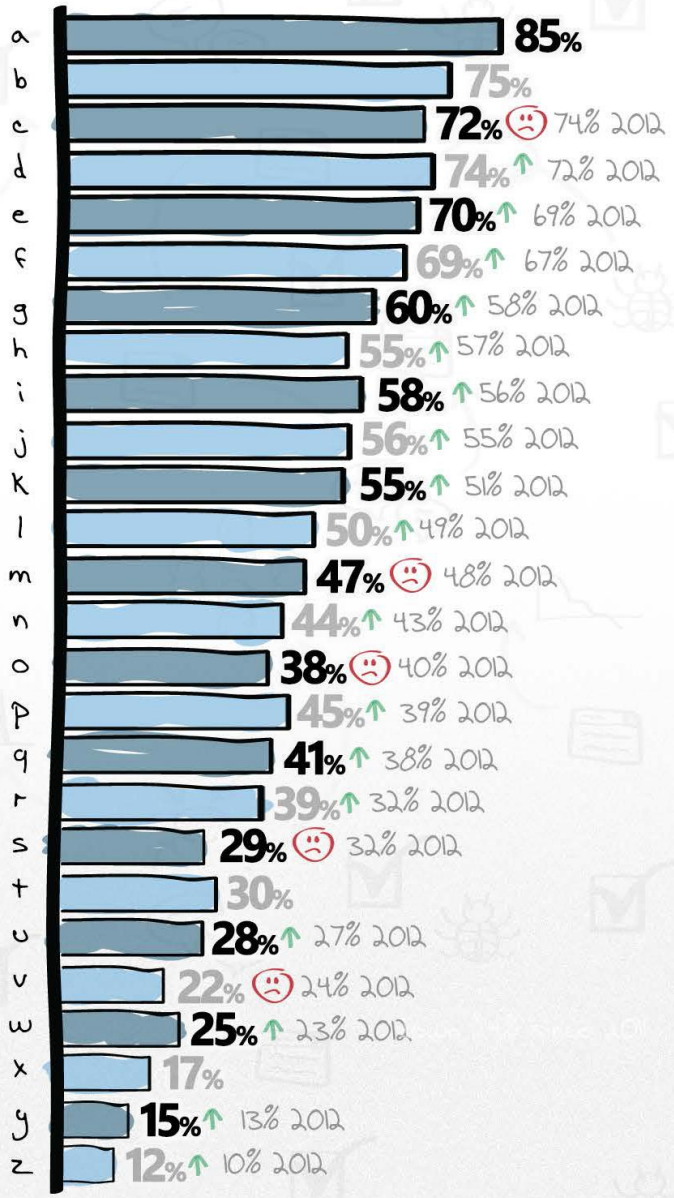
Непрерывная интеграция (англ. *Continuous Integration*) — это практика разработки программного обеспечения, которая заключается в выполнении частых автоматизированных сборок проекта для скорейшего выявления и решения интеграционных проблем. В обычном проекте, где над разными частями системы разработчики трудятся независимо, стадия интеграции является заключительной. Она может непредсказуемо задержать окончание работ. Переход к непрерывной интеграции позволяет снизить трудоёмкость интеграции и сделать её более предсказуемой за счет наиболее раннего обнаружения и устранения ошибок и противоречий.

AGILE TECHNIQUES EMPLOYED

Respondents are making use of a wide variety of different agile management techniques. More than 85% practice Daily Standups, ¾ are using Iteration Planning and Retrospectives, and nearly the same proportion said they maintain Burndown charts. Over the last 2 years we've seen a 10% increase in the use of Retrospectives (from 64% in 2011 to 74% in 2013).

*Respondents were able to select multiple options.

- a Daily Standup
- b Iteration Planning
- c Unit Testing
- d Retrospectives
- e Release Planning
- f Burndown/ Team-Based Estimation
- g Velocity
- h Coding Standards
- i Continuous Integration
- j Automated Builds
- k Dedicated Product Owner
- l Integrated Dev/QA
- m Refactoring
- n Open Workarea
- o TDD
- p Digital Taskboard
- q Story Mapping
- r Kanban
- s Collective Code Ownership
- t Pair Programming
- u Automated Acceptance Testing
- v Analog Taskboard
- w Continuous Deployment
- x Agile Games
- y Cycle Time
- z BDD



Scrum

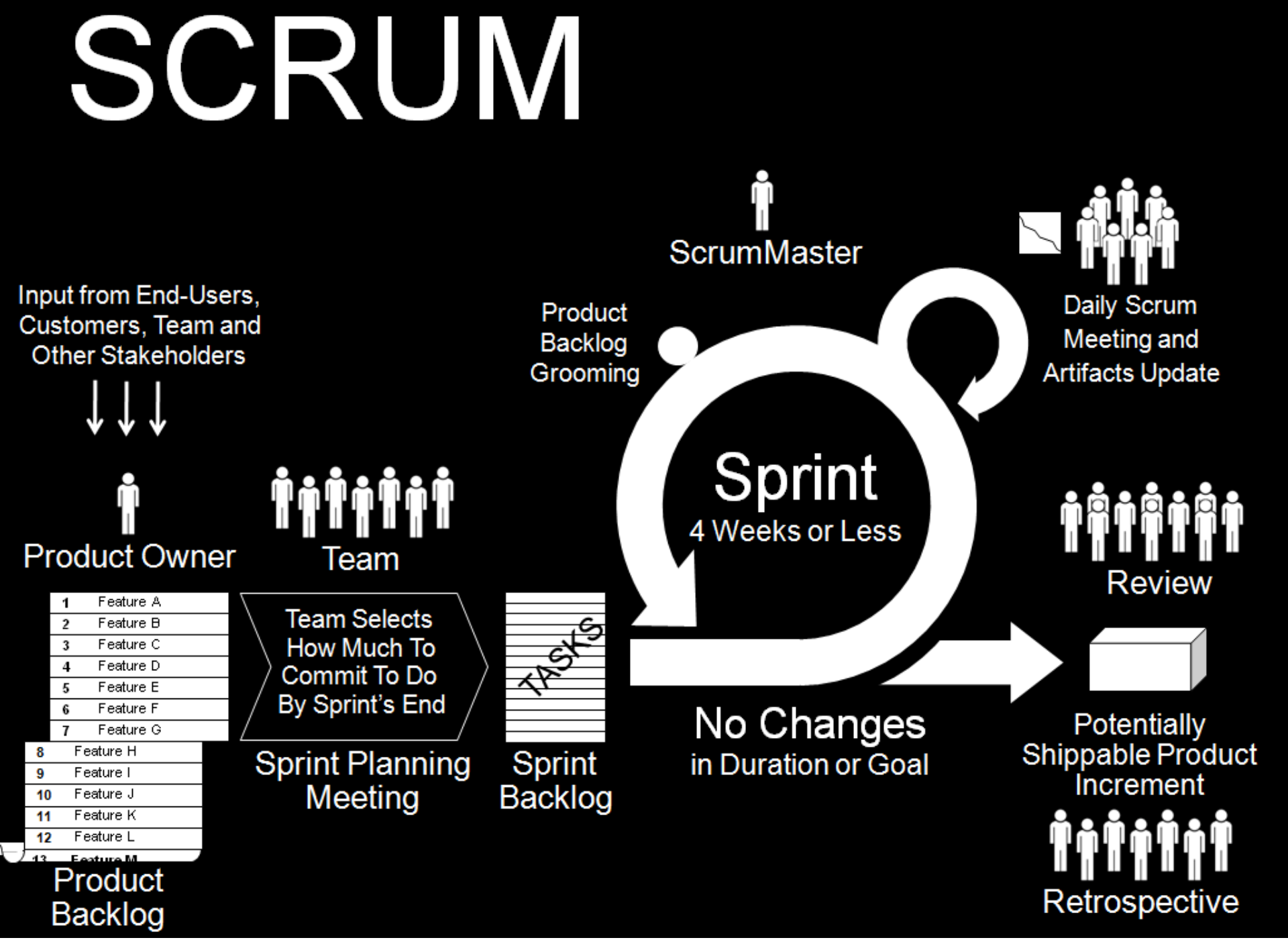
Скрам (Scrum) — это набор принципов, на которых строится процесс разработки, позволяющий в жёстко фиксированные и небольшие по времени итерации, называемые спринтами (sprints), предоставлять конечному пользователю работающее ПО с новыми возможностями, для которых определён наибольший приоритет.

Роли

- **Scrum-master** – проводит совещания (Scrum meetings) следит за соблюдением всех принципов скрам, разрешает противоречия и защищает команду от отвлекающих факторов
- **Владелец продукта** (Product Owner) — представляет интересы конечных пользователей и других заинтересованных в продукте сторон.
- **Скрам-команда** (Scrum Team) — кросс-функциональная команда разработчиков проекта, состоящая из специалистов разных профилей: тестировщиков, архитекторов, аналитиков, программистов и т. д. Размер команды в идеале составляет 7±2 человека.

- **Пользователи** (*Users*)
- **Клиенты, Продавцы (*Stakeholders*)** — лица, которые инициируют проект и для кого проект будет приносить выгоду. Они вовлечены в скрам только во время **обзорного совещания по спринту** (Sprint Review).
- **Управляющие (*Managers*)** — люди, которые управляют персоналом.
- **Эксперты-консультанты (*Consulting Experts*)**

SCRUM



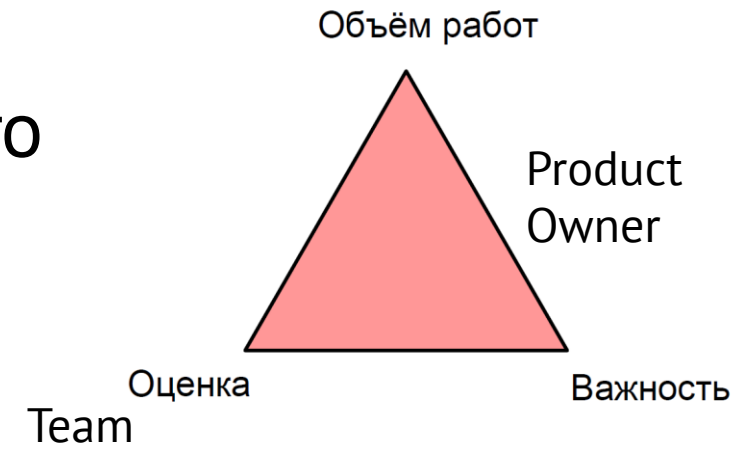
http://gurtejpsingh.wordpress.com/tag/scrum-roles/

Product backlog (резерв проекта)

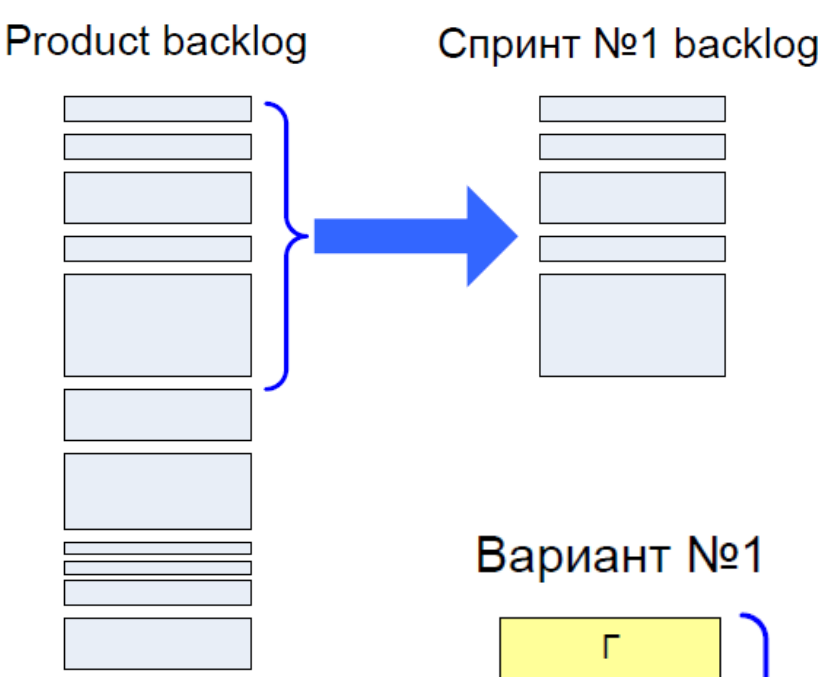
- Элементы – «истории» (user story)
  - ID
  - Название
  - Важность (int)
  - Начальная оценка объема работ (в *относительных* story point’ах, int)
  - Как продемонстрировать?
  - Компоненты
  - Категория (оптимизация, панель управления)

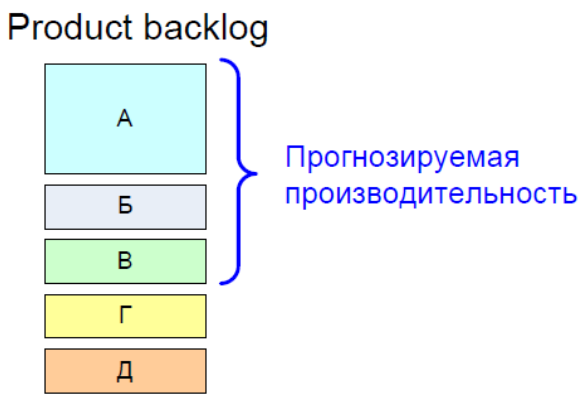
Планирование спринта

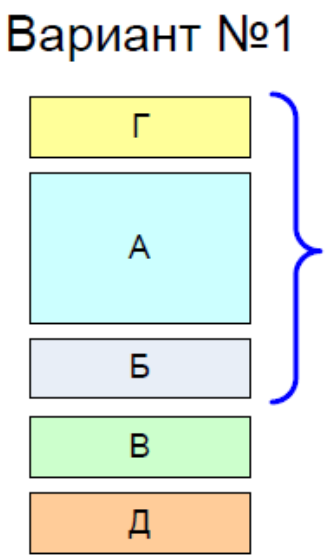
- Цель спринта
- Список участников команды (степень занятости)
- Sprint backlog (решение каждой задачи не должно занимать более 12 часов или одного дня)
- Дата демонстрации
- Место и время ежедневного scrum
- Участники: команда, владелец проекта

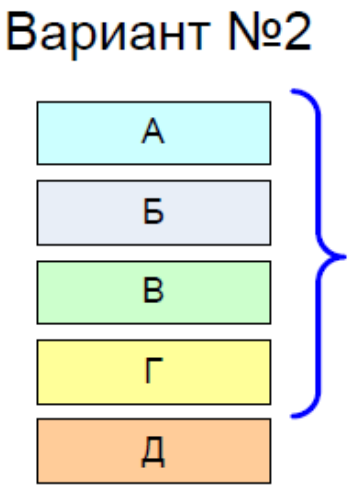


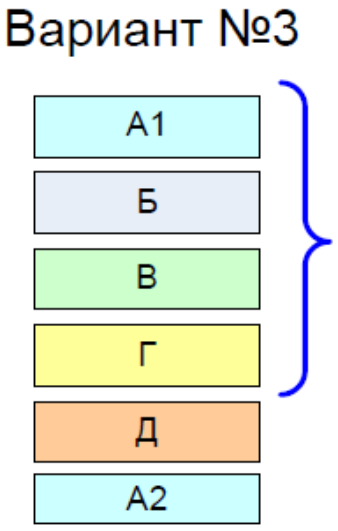
Sprint backlog



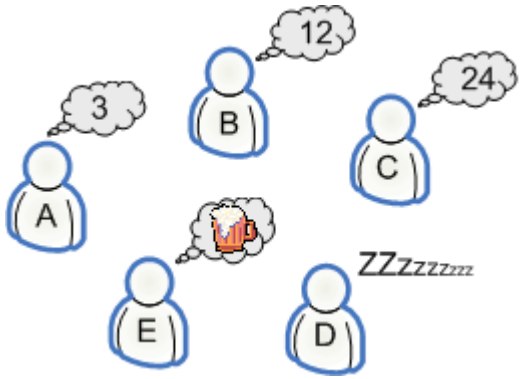


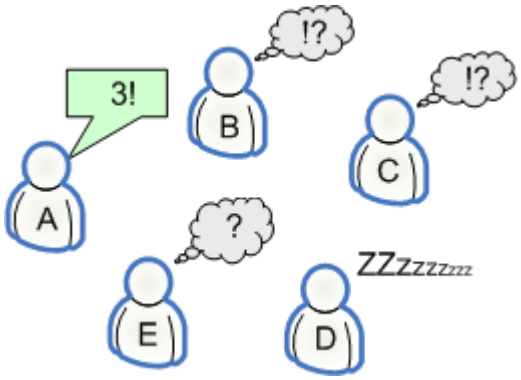


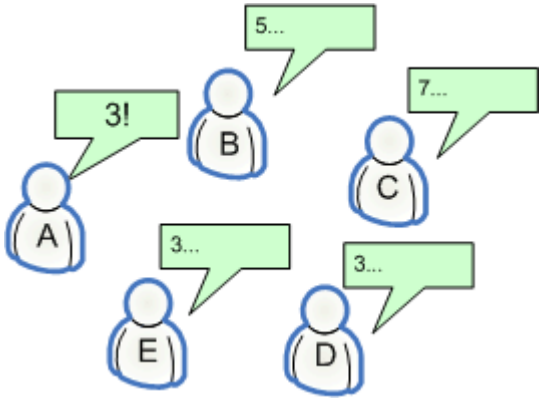




Planning poker







Vs

Ежедневное совещание

- начинается точно вовремя;
- все могут наблюдать, но только «свиньи» говорят;
- длится не более 15 минут;
- проводится в одном и том же месте в течение спринта.

В течение совещания каждый член команды отвечает на 3 вопроса:

• Что сделано с момента предыдущего ежедневного совещания?

• Что будет сделано с момента текущего совещания до следующего?

• Какие проблемы мешают достижению целей спринта? (Над решением этих проблем работает *скрам мастер*. Обычно это решение проходит за рамками ежедневного совещания и в составе лиц, непосредственно затронутых данным препятствием.)

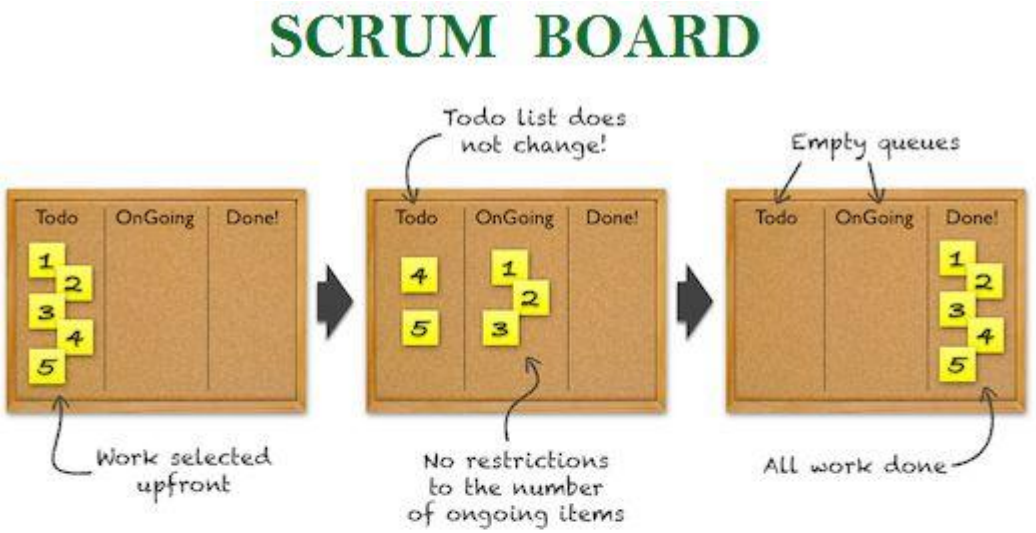
Обзор итогов спринта

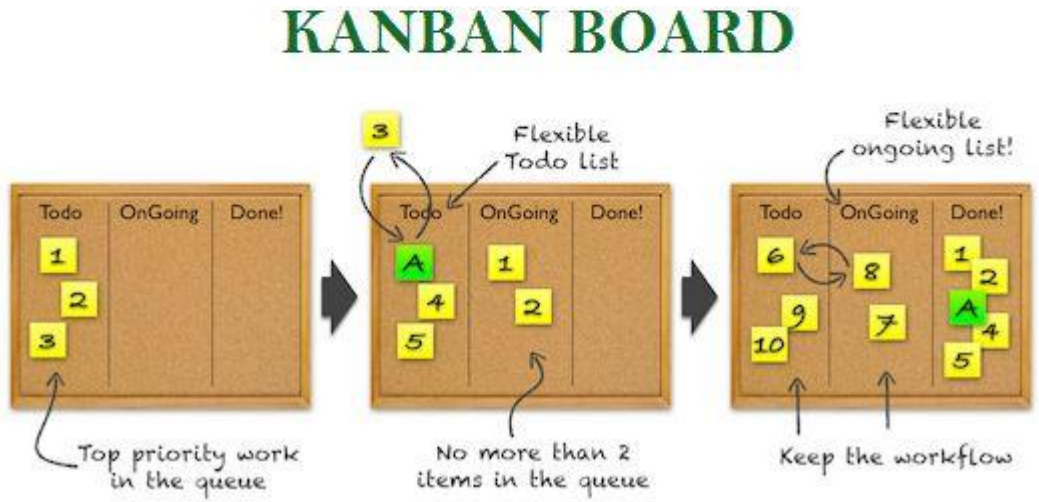
- Команда демонстрирует прирост функциональности продукта всем заинтересованным лицам.
- Привлекается максимальное количество зрителей.
- Все члены команды участвуют в демонстрации (один человек на демонстрацию или каждый показывает, что сделал за спринт).
- Нельзя демонстрировать незавершенную функциональность.
- Ограничена четырьмя часами в зависимости от продолжительности итерации и прироста функциональности продукта.

Ретроспективное совещание

- Члены команды высказывают своё мнение о прошедшем спринте.
- Отвечают на два основных вопроса:
  - Что было сделано хорошо в прошедшем спринте?
  - Что надо улучшить в следующем?
- Выполняют улучшение процесса разработки (решают вопросы и фиксируют удачные решения).
- Ограничена одним—тремя часами.

Scrum/kanban board





https://www.outsystems.com/blog/2010/11/scrum-vs-kanban.html

То, чем сегодня никто не занимается

То, чем сегодня кто-то занимается

То, чем больше никто не будет заниматься

Ставьте маркером новую точку на burndown-диаграмме каждый день после ежедневного Scrum'a

В планах | В процессе | Готово! :о)

Цель спринта | Выпуск бета-версии!

В начале стикеры задач перемещаются в этом направлении

Потом белая карточка истории перемещается в "Готово"

Рефакто-ринг 1д

Интегра-ционное

DAO 1д

Автомат. обновление

Приёмо чные тесты 2д

Покопаться в Tapesty 2д

Спец. для GUI 2д

Написать скрипт 8д

Админка: вход

Приёмо чные тесты 2д

Интегра-ция с JBoss 2д

Разраб. GUI 1д

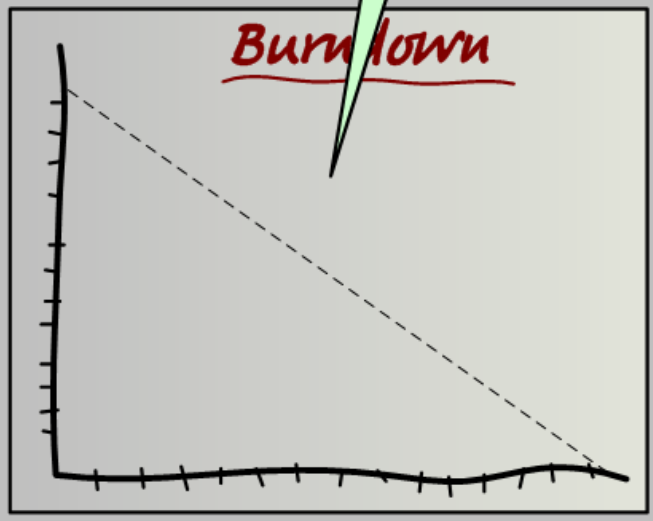
Админка: Управление пользователями

Приёмо чные тесты 3д

Дизайн для GUI (CSS) 1д

Уточнить требования 2д

Разраб. GUI 6д



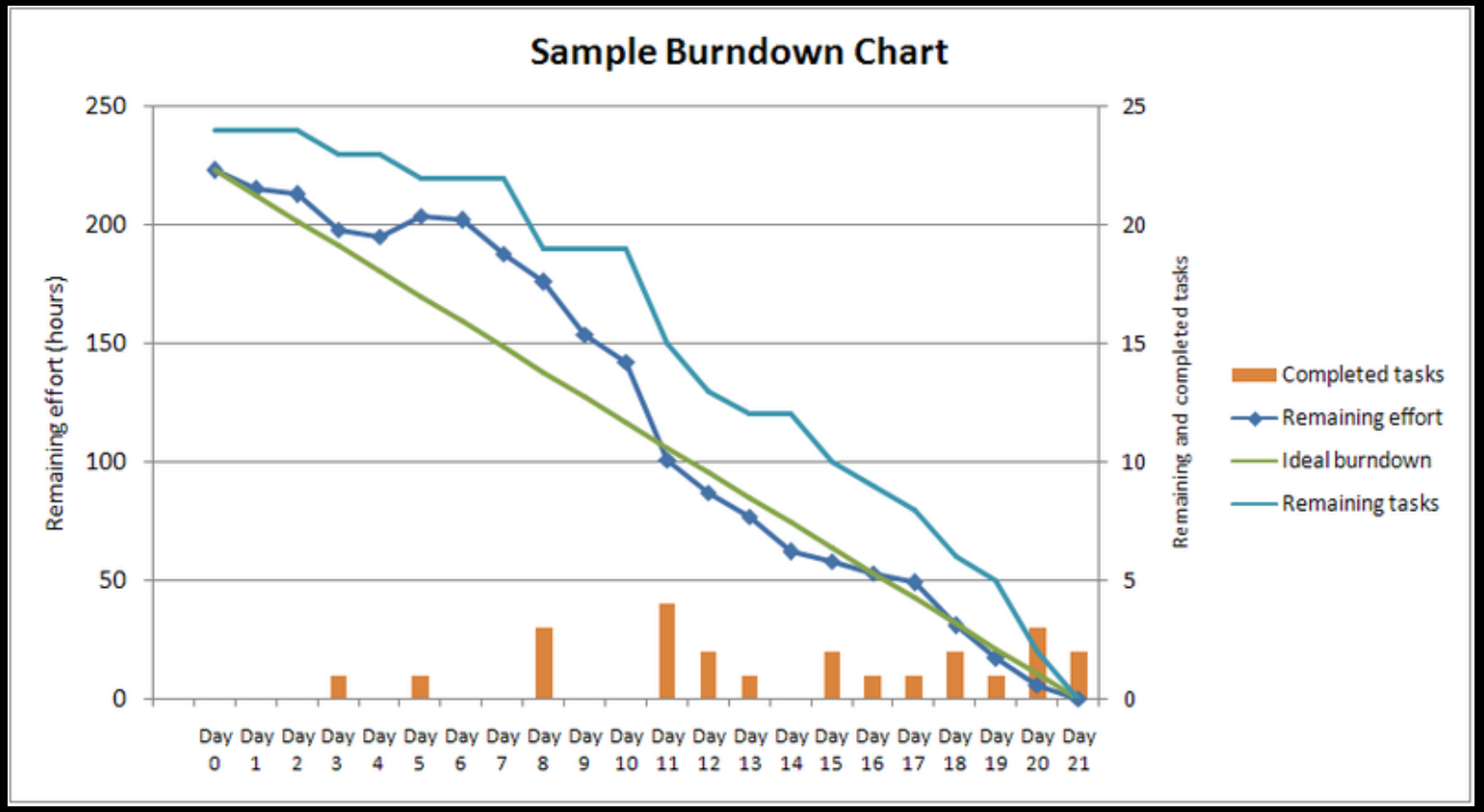
Незапланированные

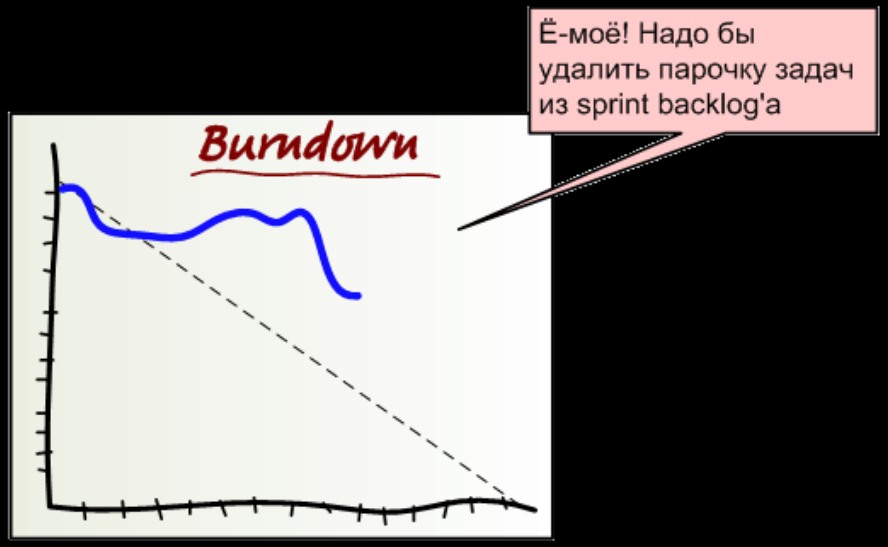
Следущие

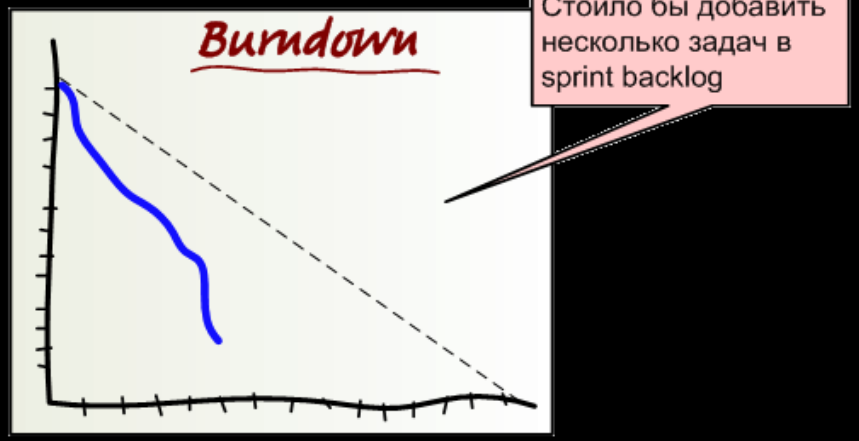
Снятие со счёта

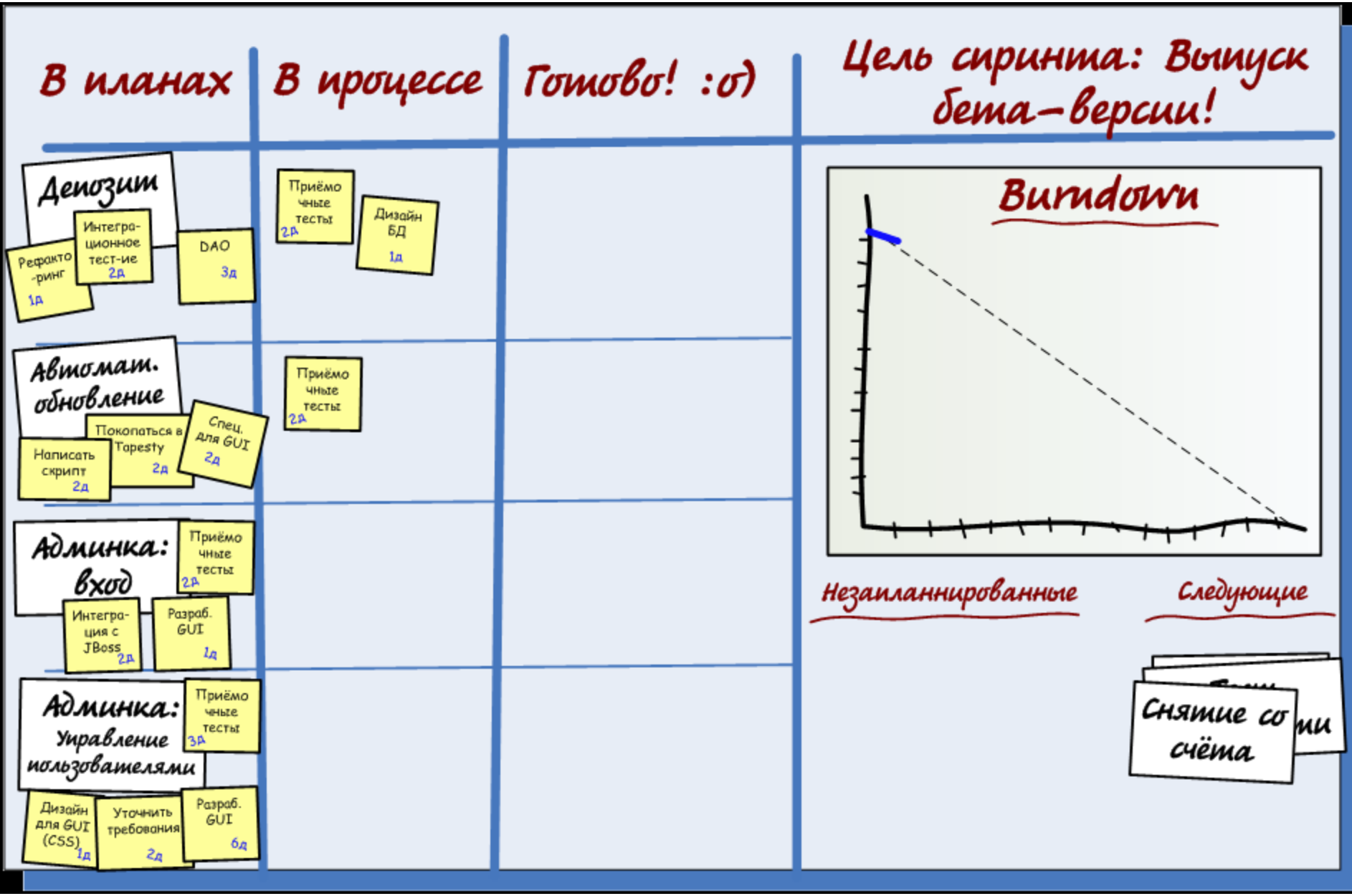
Если все истории готовы, а спринт ещё не закончился – добавь новые отсюда

http://scrum.org.ua/wp-content/uploads/2008/12/scrum_xp-from-the-trenches-rus-final.pdf

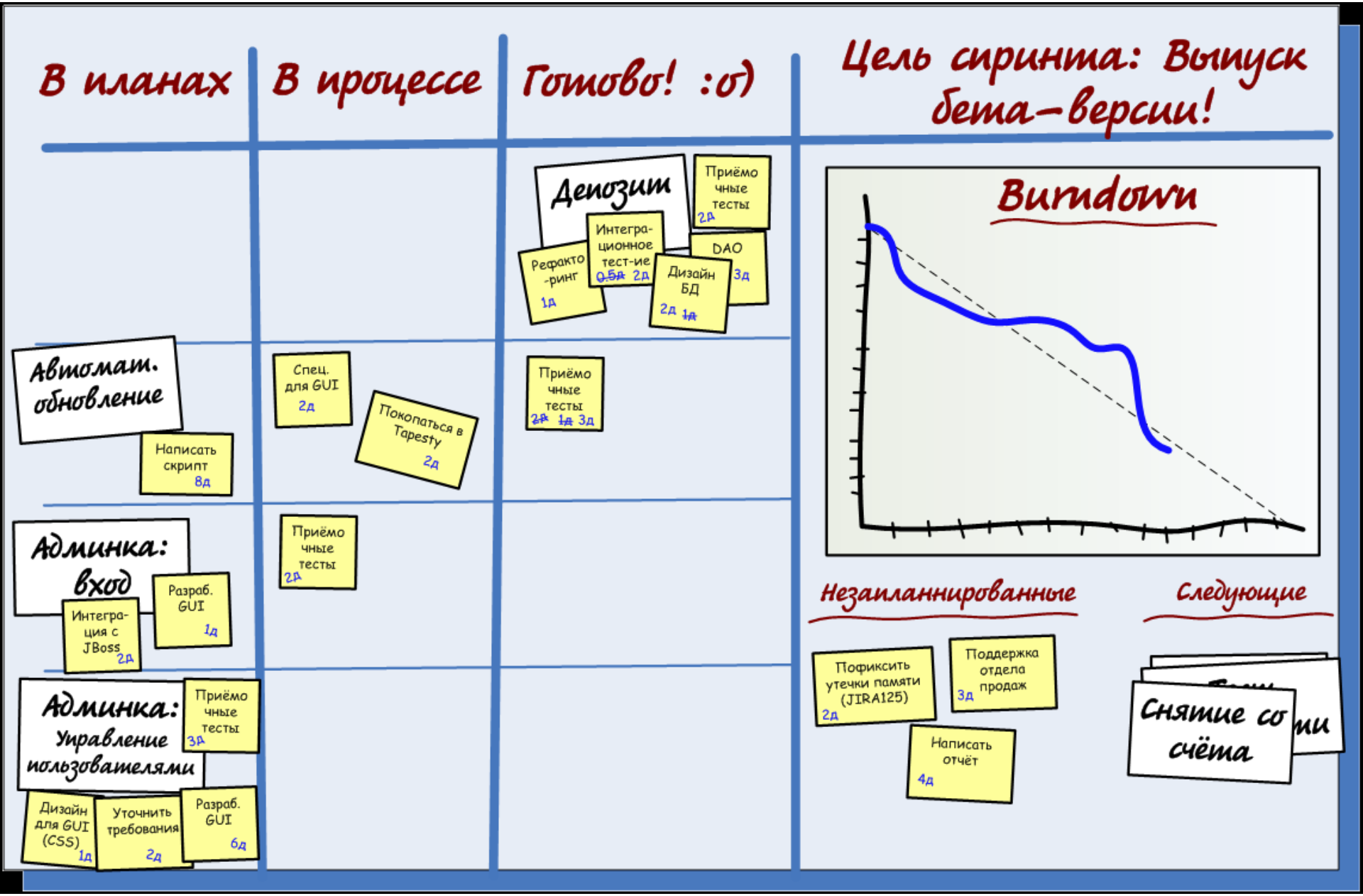

Sample Burndown Chart
Remaining effort (hours)
Remaining and completed tasks
250
200
150
100
50
0
25
20
15
10
5
0
Completed tasks
Remaining effort
Ideal burndown
Remaining tasks
Day 0
Day 1
Day 2
Day 3
Day 4
Day 5
Day 6
Day 7
Day 8
Day 9
Day 10
Day 11
Day 12
Day 13
Day 14
Day 15
Day 16
Day 17
Day 18
Day 19
Day 20
Day 21


Burndown
Ё-моё! Надо бы удалить парочку задач из sprint backlog'а


Burndown
Стоило бы добавить несколько задач в sprint backlog



http://scrum.org.ua/wp-content/uploads/2008/12/scrum_xp-from-the-trenches-rus-final.pdf

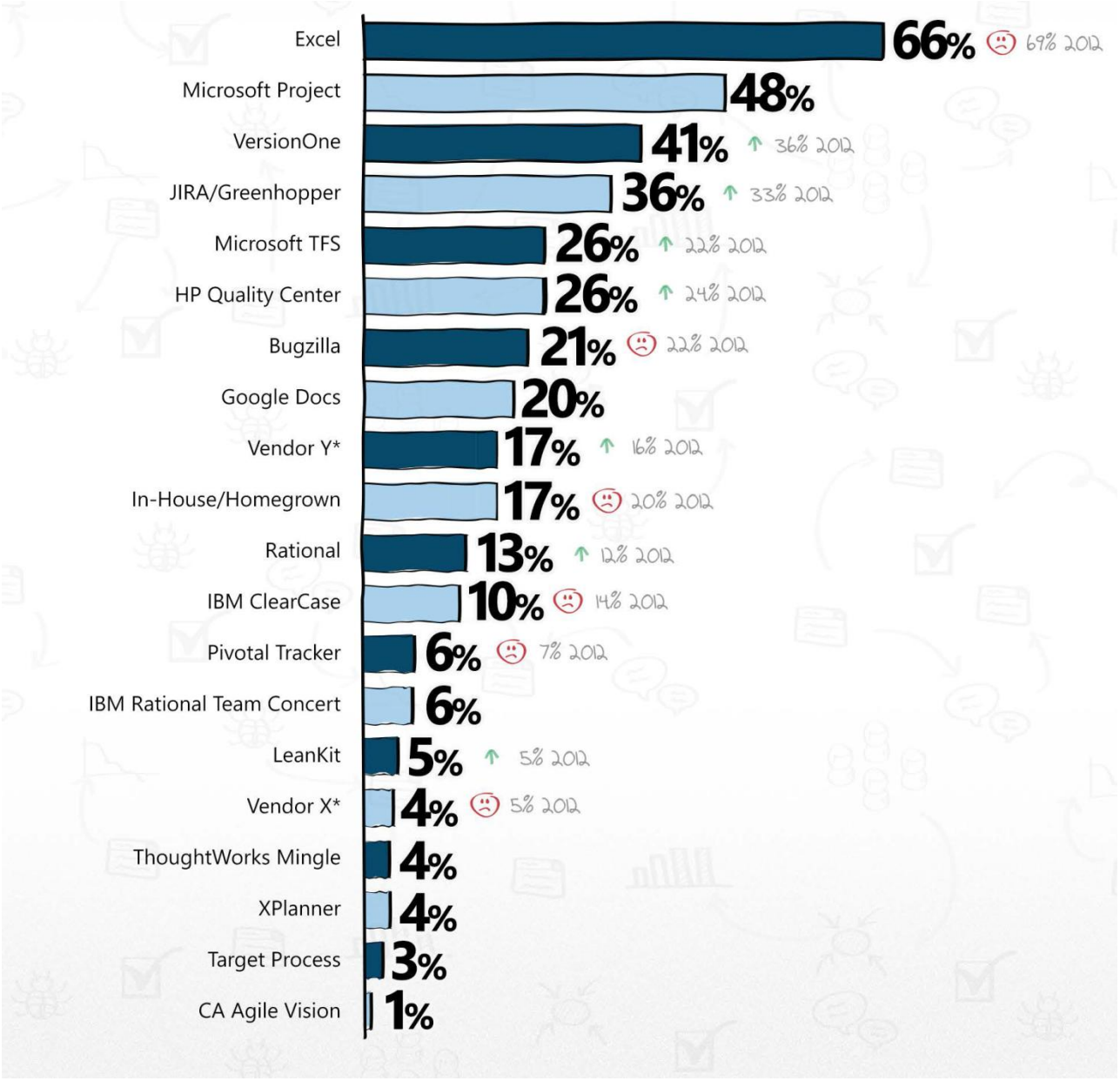


http://scrum.org.ua/wp-content/uploads/2008/12/scrum_xp-from-the-trenches-rus-final.pdf


TOTALS
To Do
NEXT
WIP
DONE
"Development"
Total : 110
Done : 16
Not Done: 94
"CAMPAIGNS"
Total 37
Done. 23
Not Done: 14
Total :21
Done: 9
Not Done: 12
Next Sprint
"Debt"
Release
CAPACITY
COUNTDOWN
New tools notification email
7526
31%


Expedite
Goals
Project Queue
Elaboration
Discussion, Acceptance
Production
Deployment
Done!
organise ATHENA
Marketing
Instructional
Events
Spaces
Organization
Website
Done in Spring 2013

Инструменты



Источники

- **Книги**
  - Д. Г. Шопырин «Гибкие технологии разработки программного обеспечения» учебно-методическое пособие, 2007 (http://books.ifmo.ru/file/pdf/422.pdf)
  - Henrik Kniberg «Scrum and XP from the Trenches», 2007 (http://www.infoq.com/minibooks/scrum-xp-from-the-trenches)
    Версия книги на русском: «Scrum и XP: заметки с передовой» (http://scrum.org.ua/wp-content/uploads/2008/12/scrum_xp-from-the-trenches-rus-final.pdf)
- **Полезные материалы**
  - Michael James, Scrum Reference Card
    http://scrumreferencecard.com/ScrumReferenceCard.pdf
- **Классика разработки ПО**
  - Управление проектами
    - «Мифический человеко-месяц или Как создаются программные системы», Фредерик Брукс, 1975
    - «Путь камикадзе», Эдвард Йордон, 2008 (Death March, 2003)
  - Общие вопросы разработки
    - «Программист-прагматик. Путь от подмастерья к мастеру», Эндрю Хант, Дэвид Томас, 2007
    - «Совершенный код», С. Макконнелл, 2005