

Лекция 3: О жизненном цикле и методологиях

Юрий Литвинов

y.litvinov@spbu.ru

27.02.2024

Виды деятельности при разработке ПО

- ▶ Возникновение и исследование идеи
- ▶ Анализ и сбор требований (пилотный проект)
- ▶ Планирование и проектирование
- ▶ Разработка
- ▶ Отладка и тестирование
- ▶ Сдача
- ▶ Сопровождение

Жизненный цикл ПО

- ▶ Период времени от возникновения идеи до прекращения использования
- ▶ Последовательность этапов
 - ▶ Состав и последовательность работ
 - ▶ Получаемые результаты
 - ▶ Методы и средства
 - ▶ Роли и ответственности
 - ▶ ...
- ▶ Модели жизненного цикла

Самая популярная модель разработки ПО

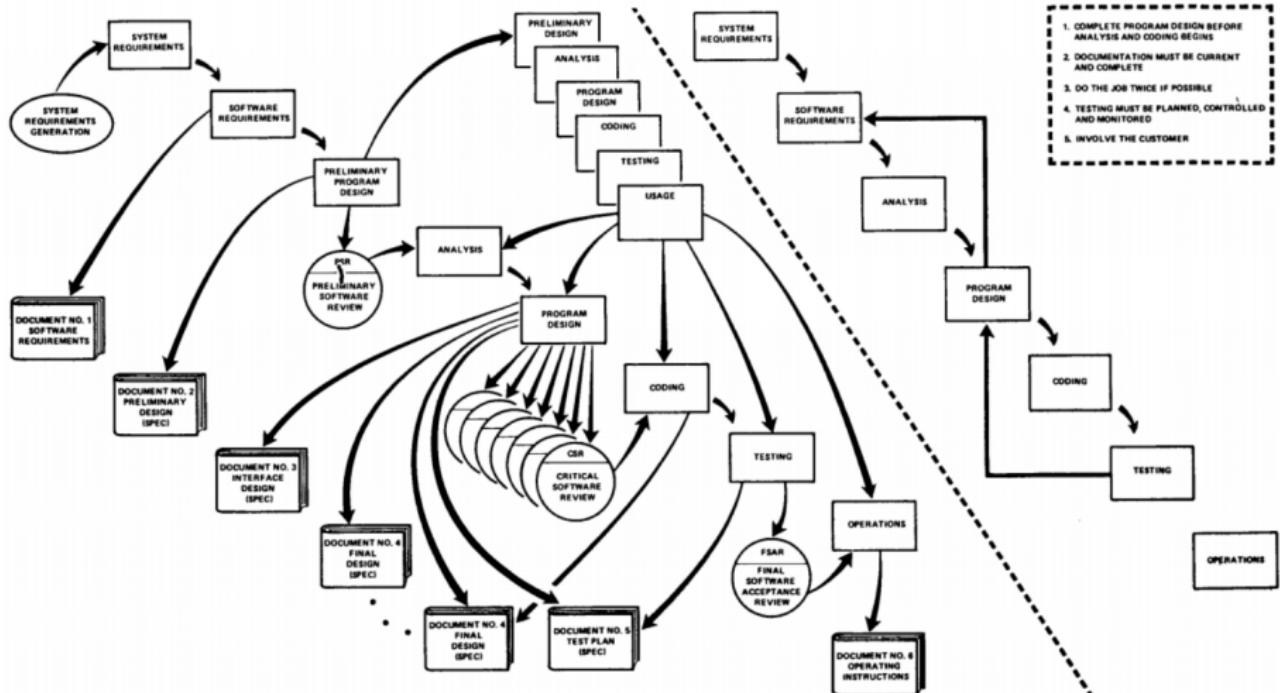


Водопадная (каскадная) модель

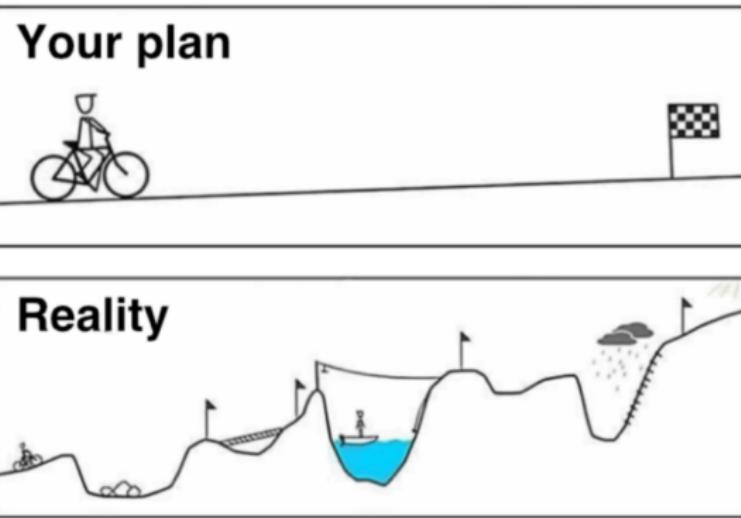


B. Ройс, 1970г, “Managing the Development of Large Software Systems”

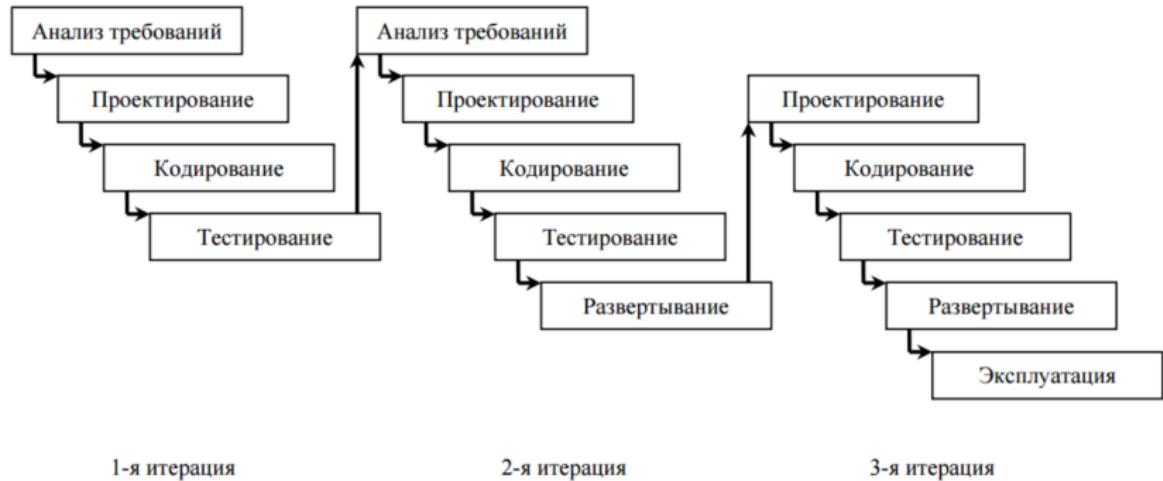
Водопадная (каскадная) модель



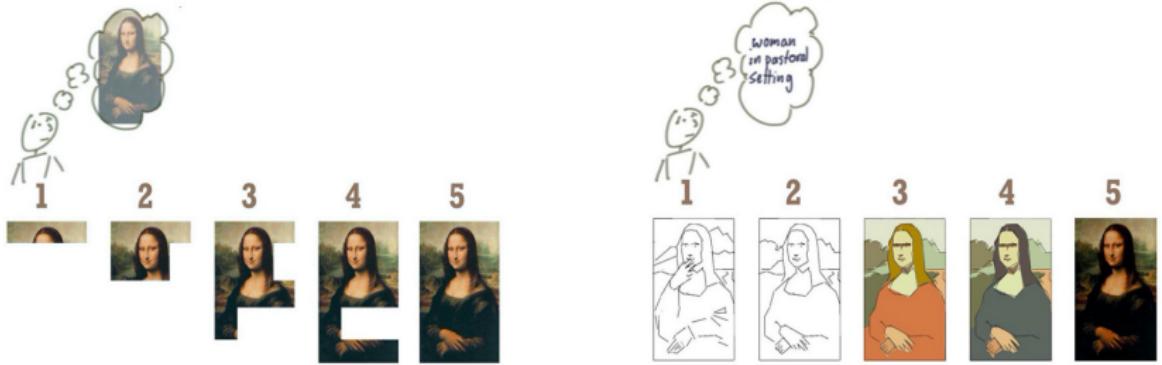
Водопадная (каскадная) модель



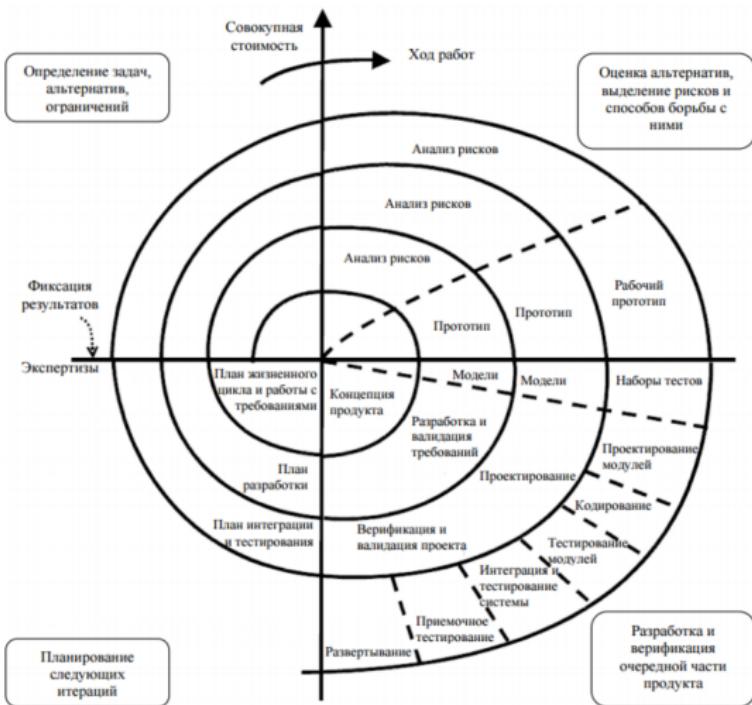
Итеративная модель



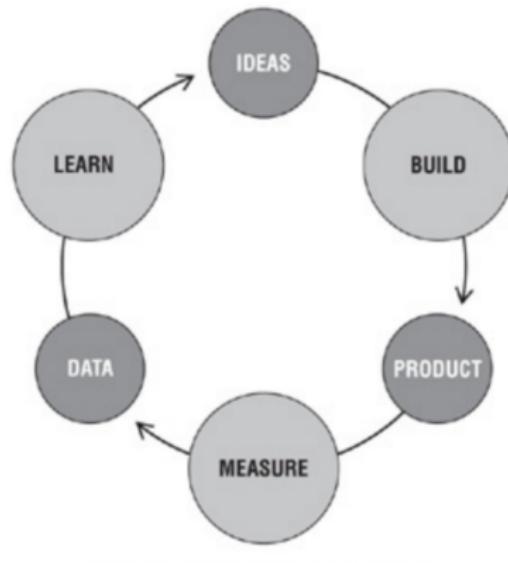
Итеративно-инкрементальная модель



Спиралевидная модель

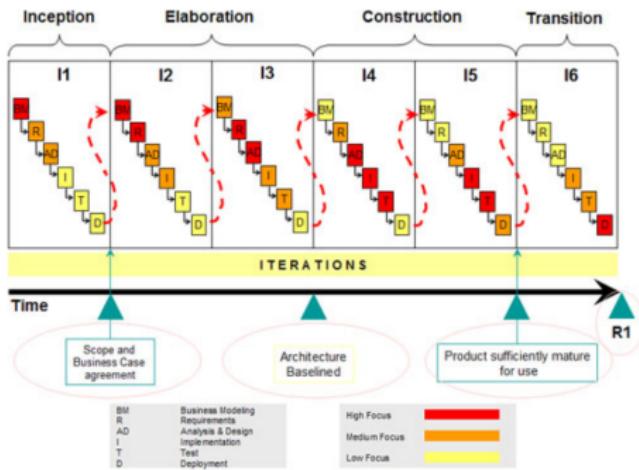


The Lean Startup Model



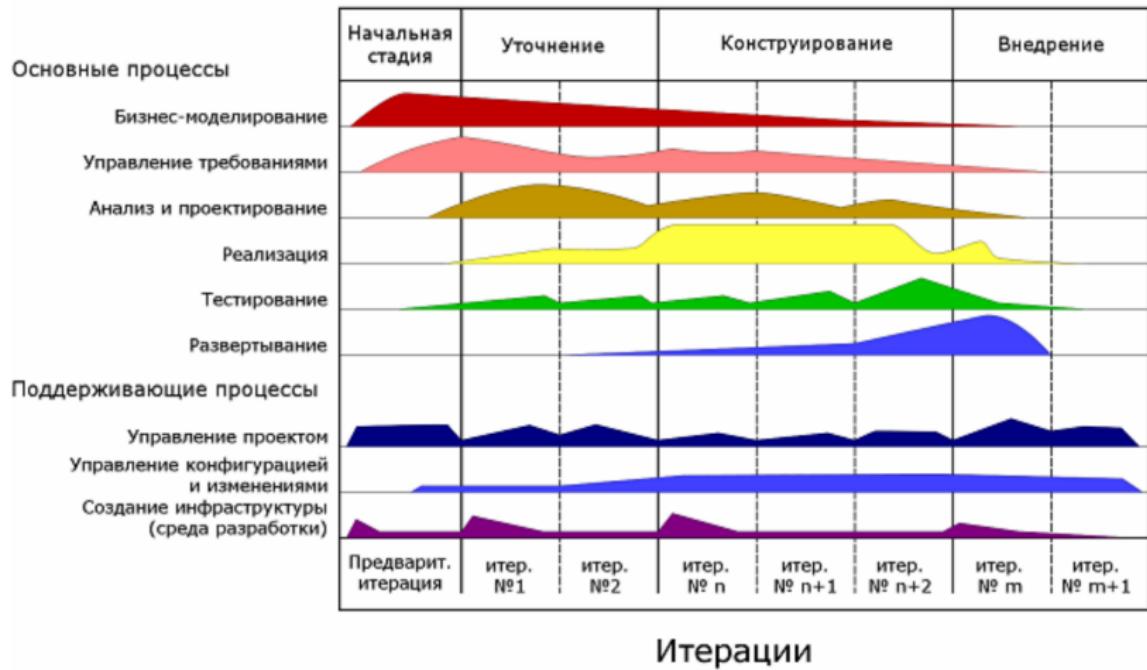
Rational Unified Process (RUP)

- ▶ Пример “тяжёлого” процесса разработки
 - ▶ Отделение практик от людей
- ▶ Ivar Jacobson, 1980-е годы
- ▶ Ericsson/Objectory AB/Rational
- ▶ Основные идеи:
 - ▶ варианты использования
 - ▶ архитектура, архитектура, архитектура
 - ▶ управляемые итерации



Рабочие процессы

Стадии



Модели, создаваемые в RUP

- ▶ Модель случаев использования
- ▶ Модель анализа (концептуальная модель)
- ▶ Модель проектирования
- ▶ Модель реализации
- ▶ Модель развёртывания
- ▶ Модель тестирования

Принципы RUP

- ▶ Выработка концепции проекта (project vision) в его начале
- ▶ Управление по плану
- ▶ Снижение рисков и отслеживание их последствий
- ▶ Тщательное экономическое обоснование всех действий
- ▶ Как можно более раннее формирование базовой архитектуры
- ▶ Использование компонентной архитектуры
- ▶ Прототипирование, инкрементная разработка и тестирование
- ▶ Регулярные оценки текущего состояния
- ▶ Управление изменениями
- ▶ Нацеленность на создание работоспособного продукта
- ▶ Нацеленность на качество
- ▶ Адаптация процесса под нужды проекта

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/manifesto.html>

12 принципов Agile (1)

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

12 принципов Agile (2)

5. Проектом занимаются мотивированные личности, которые обеспечены нужными условиями работы, поддержкой и доверием
6. Рекомендуемый метод передачи информации — личный разговор (лицом к лицу)
7. Работающее программное обеспечение — лучший измеритель прогресса
8. Инвесторы, разработчики и пользователи должны иметь возможность поддерживать постоянный темп на неопределённый срок

12 принципов Agile (3)

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.
Команда должна систематически анализировать возможные способы улучшения эффективности и соответственно корректировать стиль своей работы

eXtreme Programming (XP)

- ▶ Кент Бек, примерно 1999
- ▶ Первая широкоизвестная agile-методология
- ▶ Основные принципы: коммуникация, простота, обратная связь, храбрость
- ▶ Непопулярна сейчас
 - ▶ Слишком революционные практики

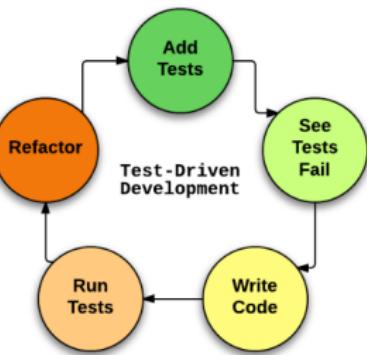
Практики XP

1. Короткий цикл обратной связи:

- ▶ Разработка через тестирование
- ▶ Игра в планирование
- ▶ Заказчик всегда рядом
- ▶ Парное программирование

1. Разработка через тестирование

- ▶ Большое число автоматических тестов
 - ▶ Модульные тесты
 - ▶ Функциональные тесты
- ▶ 100% тестов должно проходить всегда
- ▶ Тесты как документирование кода
- ▶ Тесты как основа для рефакторинга
- ▶ Написание тестов перед написанием кода



2. Игра в планирование

- ▶ Быстро получить приблизительный план работы
 - ▶ Последовательное уточнение
- ▶ Идеальное время и load factor
- ▶ Распределение ответственности между командой и заказчиком
 - ▶ Приоритизация задач
- ▶ Customer stories
 - ▶ Estimatable
 - ▶ Testable
 - ▶ Bite-sizes
 - ▶ Progress

3. Заказчик всегда рядом

- ▶ Представитель пользователей в команде
 - ▶ Принимает на себя ответственность
 - ▶ Влияет на ход разработки
- ▶ Снижение затрат на коммуникацию

4. Парное программирование

- ▶ Один компьютер, два программиста
 - ▶ Регулярное перемешивание пар
- ▶ Более хороший дизайн и код
- ▶ Повышение дисциплины
- ▶ Коллективное владение кодом
- ▶ Командный дух
- ▶ Наставничество и обучение
- ▶ Непрерывная социализация

Практики XP (2)

2. Непрерывный процесс:

- ▶ Непрерывная интеграция
- ▶ Рефакторинг
- ▶ Частые небольшие релизы

5. Непрерывная интеграция

- ▶ Частые слияния веток разработки и сборки проекта
 - ▶ Несколько раз в день
 - ▶ По внешнему запросу
 - ▶ По расписанию
 - ▶ По событию
- ▶ Использование систем версионирования кода
- ▶ Максимальная автоматизация
- ▶ Постоянное наличие рабочей версии
- ▶ Затраты на интеграцию

6. Рефакторинг

- ▶ Код постоянно меняется, и это хорошо и правильно
- ▶ Рефакторинг — средство поддержки эволюции
 - ▶ Долги проектирования
- ▶ Реализуем только то, что нужно сейчас
 - ▶ Решаем проблемы по мере поступления
- ▶ Нужно много хороших тестов



Джон Томпсон, шляпных дел мастер,
изготавливает и продает шляпы
за наличный расчет

Джон Томпсон

7. Частые небольшие релизы

- ▶ Меньше функциональности
 - ▶ Проще разрабатывать и тестировать
- ▶ Быстрее доставка фич заказчику
- ▶ Быстрее обратная связь от заказчика
- ▶ Отказ от сдвига релизов
 - ▶ Урезаем функциональность

Практики XP (3)

3. Понимание, разделяемое всеми:

- ▶ Простота архитектуры
- ▶ Метафора системы
- ▶ Коллективное владение кодом
- ▶ Стандарт кодирования

8. Простота архитектуры

- ▶ Аллергия на BDUF¹
- ▶ Непрерывное проектирование
- ▶ Самые простые решения
 - ▶ Ориентация на текущие задачи
 - ▶ Простой не значит плохой
 - ▶ Когда нельзя ничего выкинуть, чтобы не потерять функциональность
 - ▶ Непрерывный рефакторинг в зависимости от условий
- ▶ Отказ от долгосрочного планирования

¹Big Design Up-Front

9. Метафора системы

- ▶ Единое описание того, как работает система
 - ▶ Текстовое описание, понятное всем
- ▶ Замена общепринятой архитектуры

10. Коллективное владение кодом

- ▶ Разделение ответственности за весь код
 - ▶ Автор кода — вся команда
- ▶ Каждый может менять любой код
 - ▶ Высокий bus factor
 - ▶ Высокие требования к команде

11. Стандарт кодирования

- ▶ Единый стандарт написания кода
 - ▶ Не тратим время на споры о несущественном
- ▶ Простота изменений, интеграции, рефакторинга
- ▶ Автоматизация проверок стиля

12. 40-часовая рабочая неделя

- ▶ Авралы и переработки вредны на перспективе
 - ▶ Работоспособность падает
 - ▶ Выгорание
- ▶ Жертвенность на работе — признак непрофессионализма

1,857 contributions in 2016

Contribution settings ▾



1,673 contributions in 2017

Contribution settings ▾



1,224 contributions in 2018

Started burning out

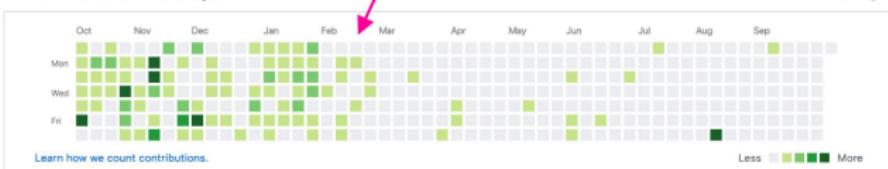
Contribution settings ▾

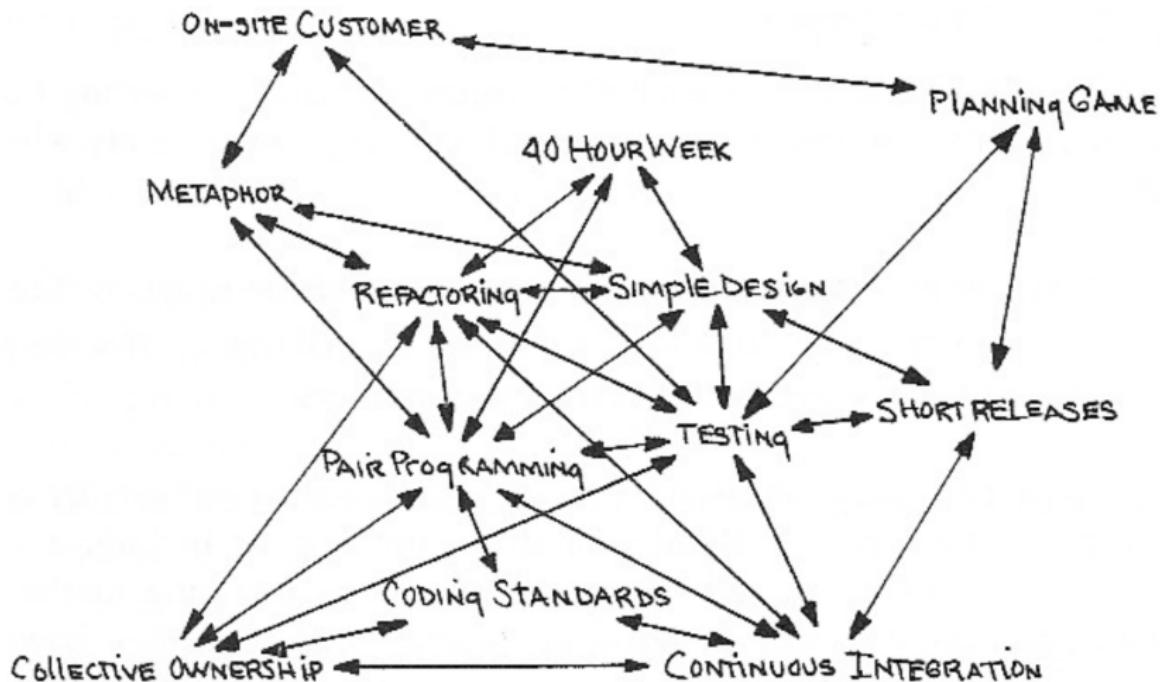


287 contributions in the last year

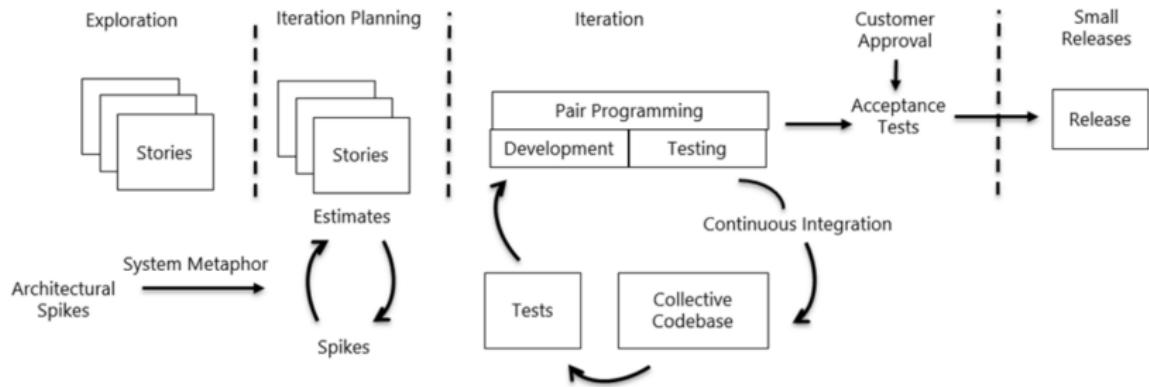
Fully burnt out

Contribution settings ▾

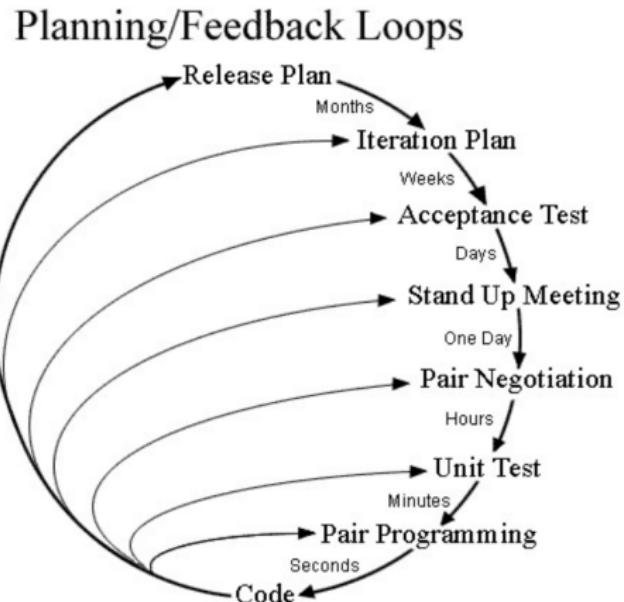




Процесс XP



Итерации в ХР



XP: резюме

- ▶ Применимость
 - ▶ Небольшие и средние команды
 - ▶ Неясные или быстро меняющиеся требования
- ▶ Повышение доверия заказчика к программному продукту
- ▶ Минимизация ошибок на ранних стадиях
- ▶ Сокращение сроков разработки
- ▶ Повышение прогнозируемости
- ▶ Повышение качества ПО

Критика XP

- ▶ Высокие требования команде
- ▶ Наличие заказчика в команде
- ▶ Project scope creep
- ▶ Постоянная переработка кода
- ▶ Требования в виде приемочных тестов
- ▶ Отсутствие целостного дизайна
- ▶ Парное программирование

