Software Engineering

Лекция 02: О жизненном цикле и процессе разработки ПО

Тимофей Брыксин
timofey.bryksin@gmail.com

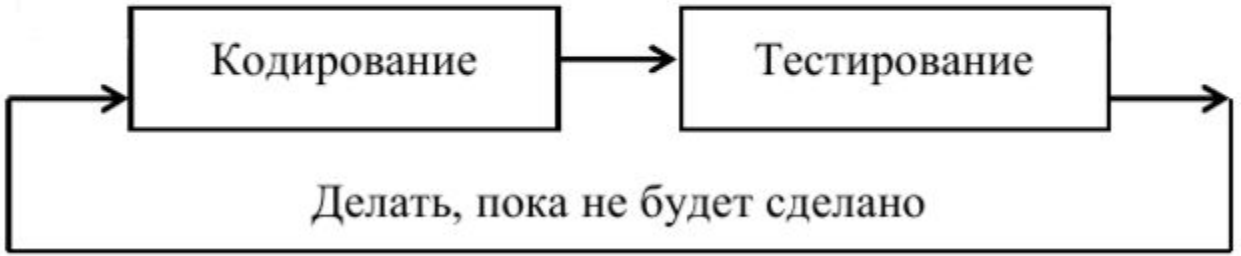
Виды деятельности при разработке ПО

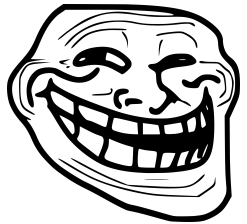
- возникновение и исследование идеи
- анализ и сбор требований (пилотный проект)
- планирование и проектирование
- разработка
- отладка и тестирование
- сдача
- сопровождение

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

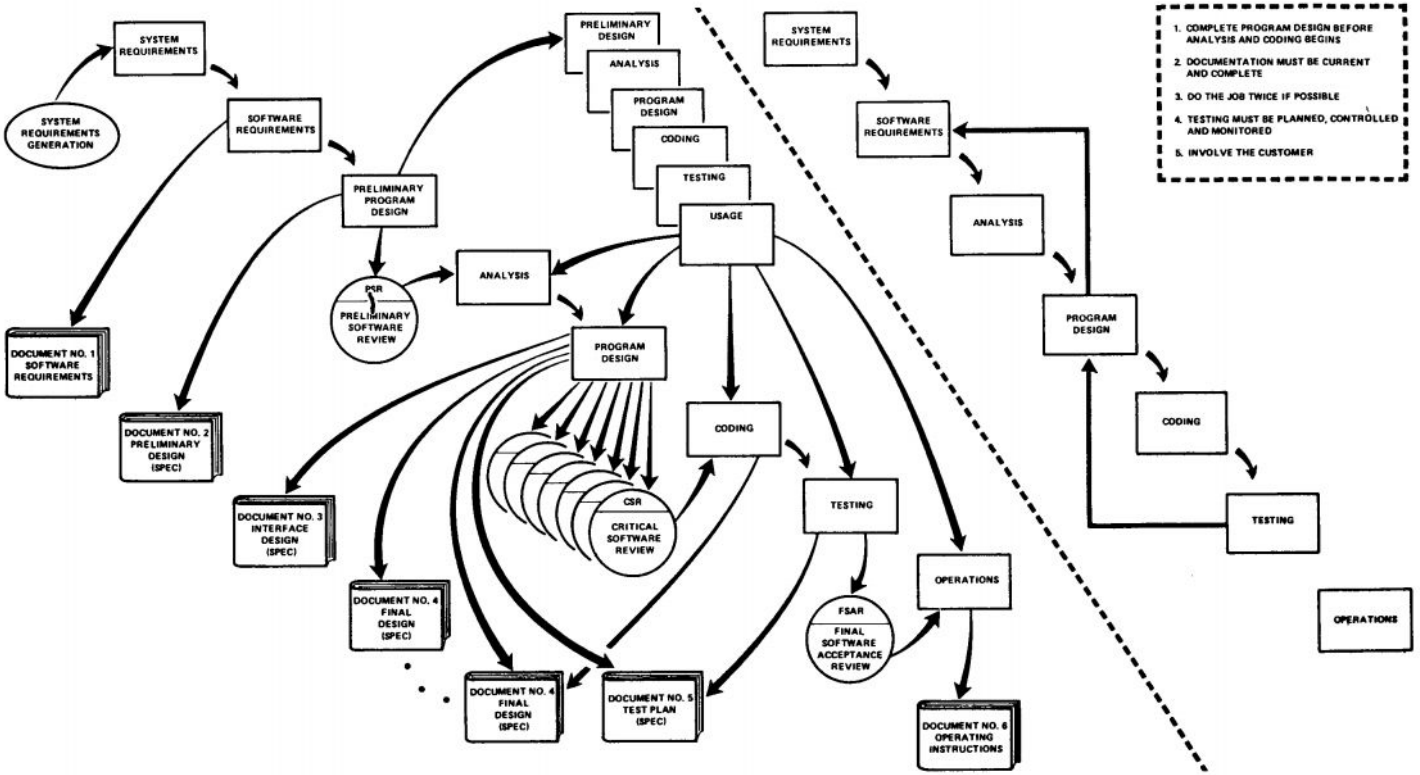
Самая популярная модель разработки ПО



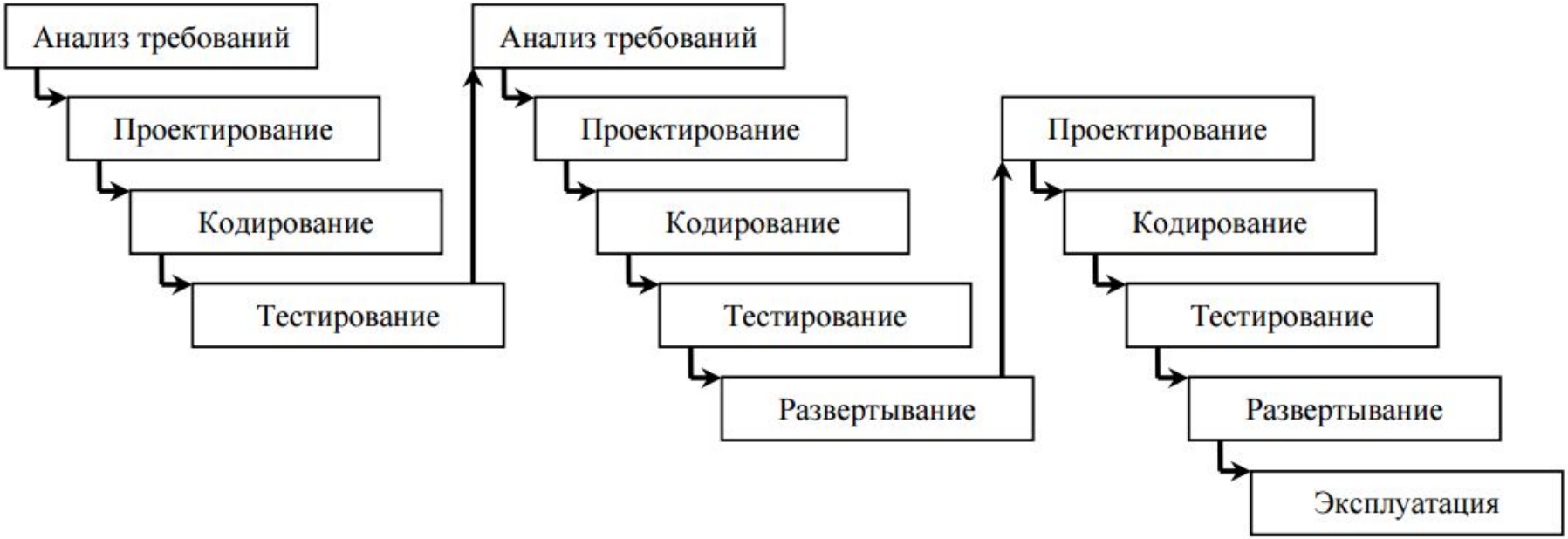


Водопадная (каскадная) модель

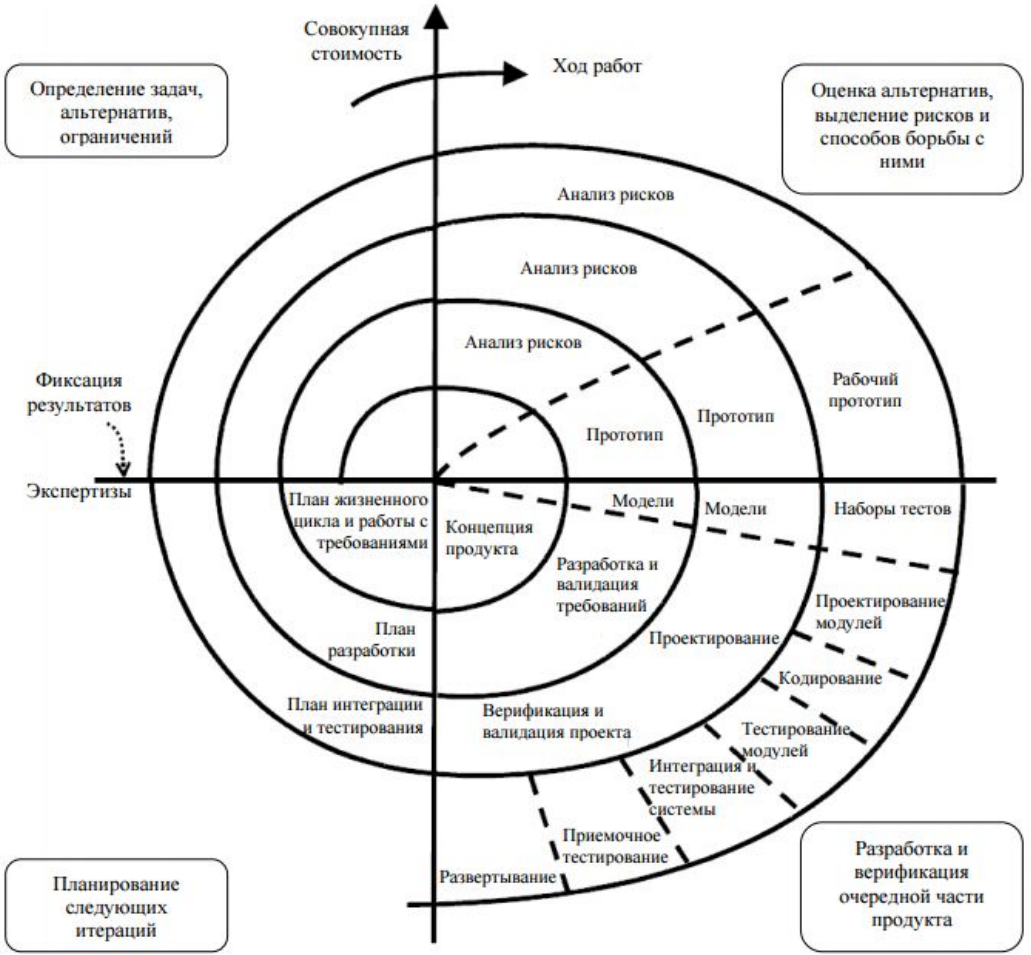
Водопадная (каскадная) модель



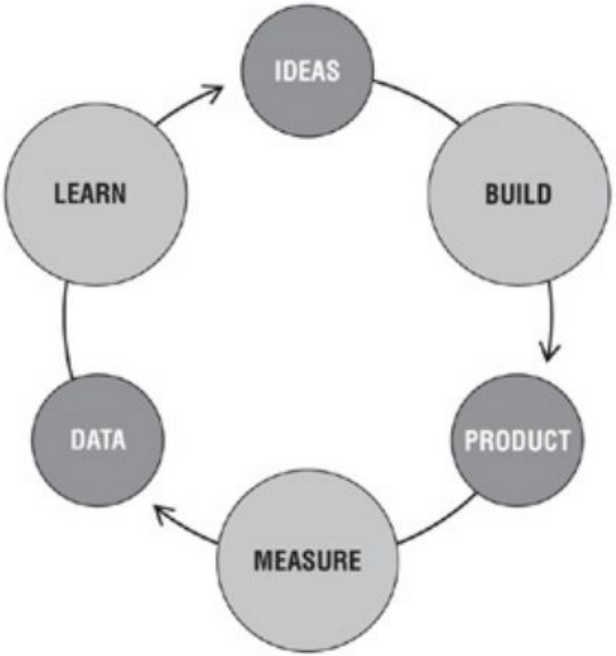
Итеративная модель



Спиральная модель



The Lean Startup Model



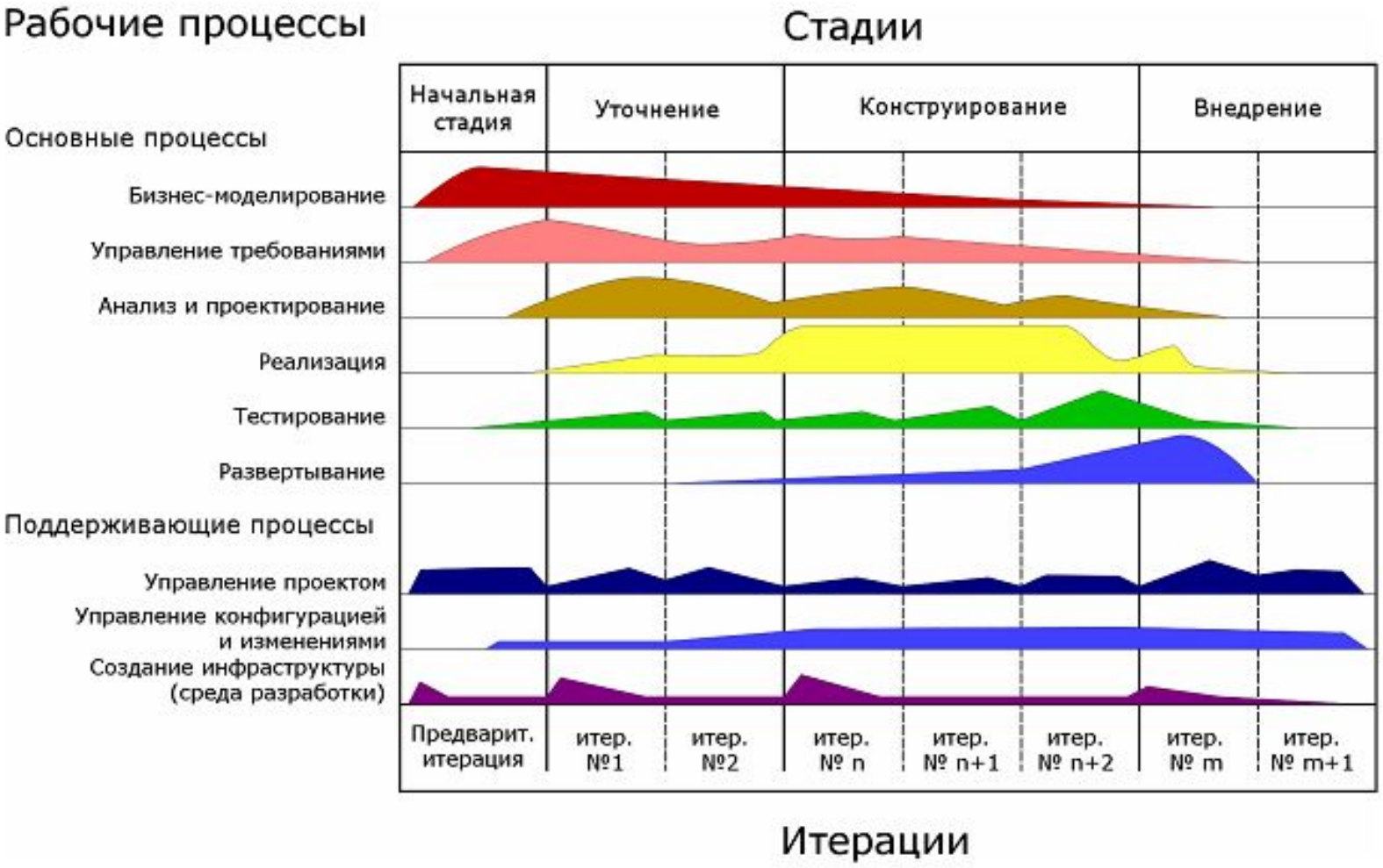
Minimize *TOTAL* time through the loop

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

Модели, создаваемые в RUP

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


Рабочие процессы
Стадии
Начальная стадия
Уточнение
Конструирование
Внедрение
Основные процессы
Бизнес-моделирование
Управление требованиями
Анализ и проектирование
Реализация
Тестирование
Развертывание
Поддерживающие процессы
Управление проектом
Управление конфигурацией и изменениями
Создание инфраструктуры (среда разработки)
Предварит. итерация
итер. №1
итер. №2
итер. № n
итер. № n+1
итер. № n+2
итер. № m
итер. № m+1
Итерации

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.

12 принципов Agile

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

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

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

- коммуникация
- простота
- обратная связь
- храбрость

Практики XP

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)

Непрерывный процесс

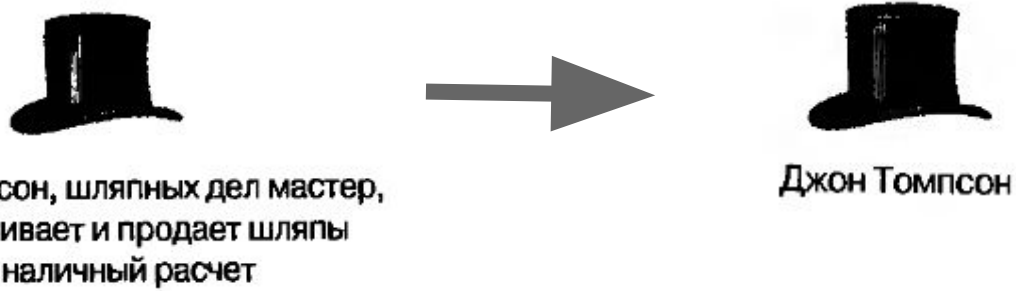
- Непрерывная интеграция
- Рефакторинг
- Частые небольшие релизы

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
- как хорошо, что есть тесты!



7. Частые небольшие релизы

- меньше функциональности
  - проще разрабатывать и тестировать
- быстрее доставка фич заказчику
- быстрее feedback от заказчика
- отказ от сдвига релизов
  - урезаем функциональность

Практики XP (3)

Понимание, разделяемое всеми

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

8. Простота архитектуры

- аллергия на BDUF
- 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

9. Метафора системы

- единое описание того, как работает система
  - текстовое описание, понятное всем
- замена общепринятой архитектуры

10. Коллективное владение кодом

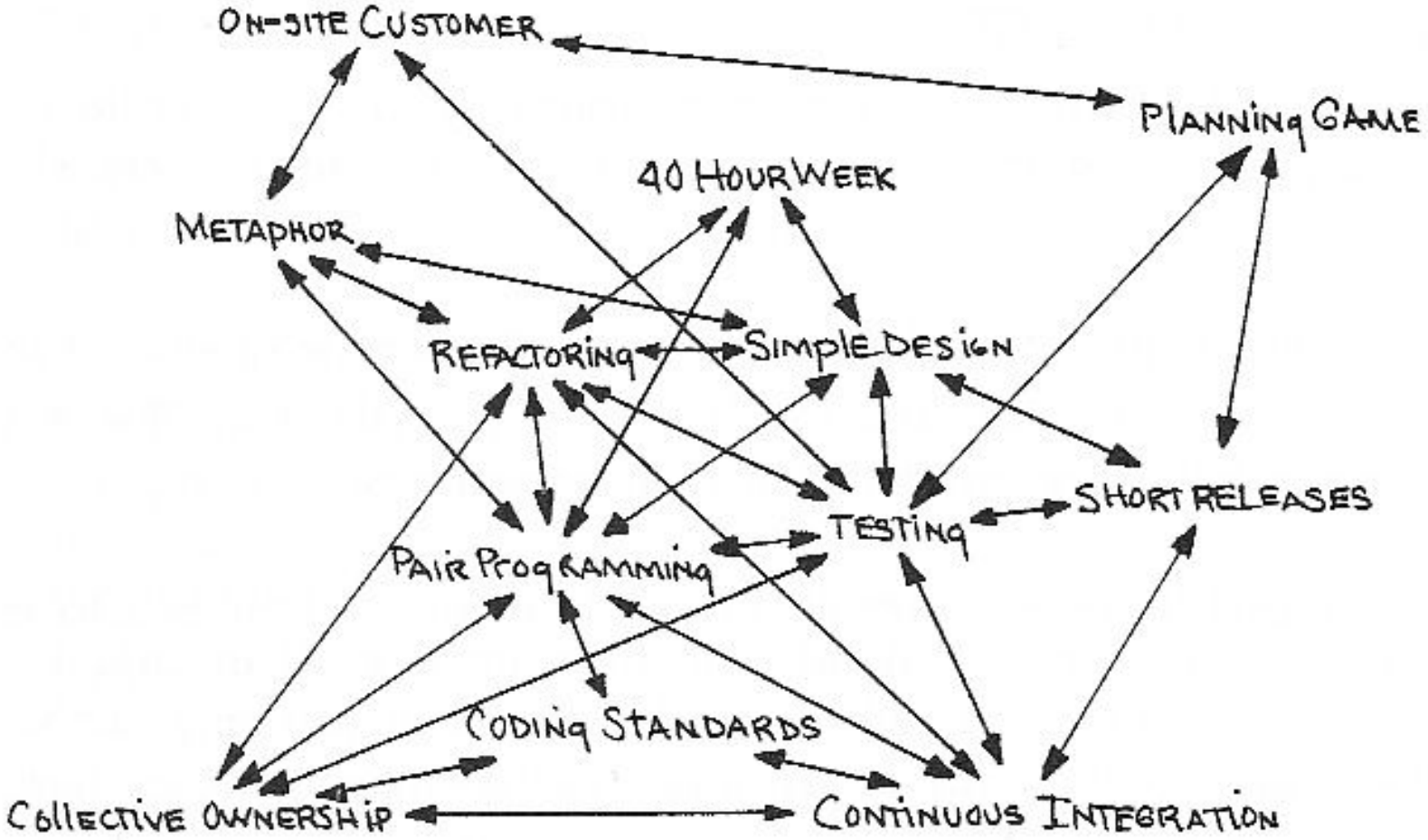
- разделение ответственности за весь код
  - автор кода -- вся команда
- каждый может менять любой код
  - высокий bus factor
  - высокие требования к команде

11. Стандарт кодирования

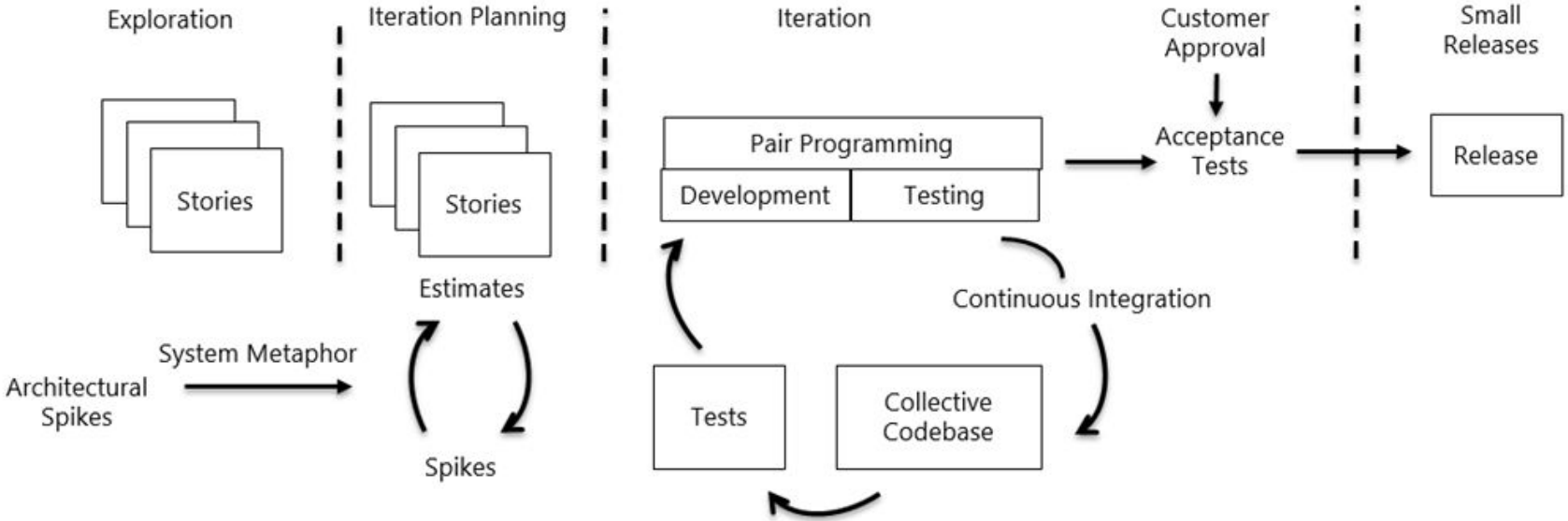
- единый стандарт написания кода
  - не тратим время на споры о несущественном
- простота изменений, интеграции, рефакторинга
- автоматизация проверок стиля

12. 40-часовая рабочая неделя

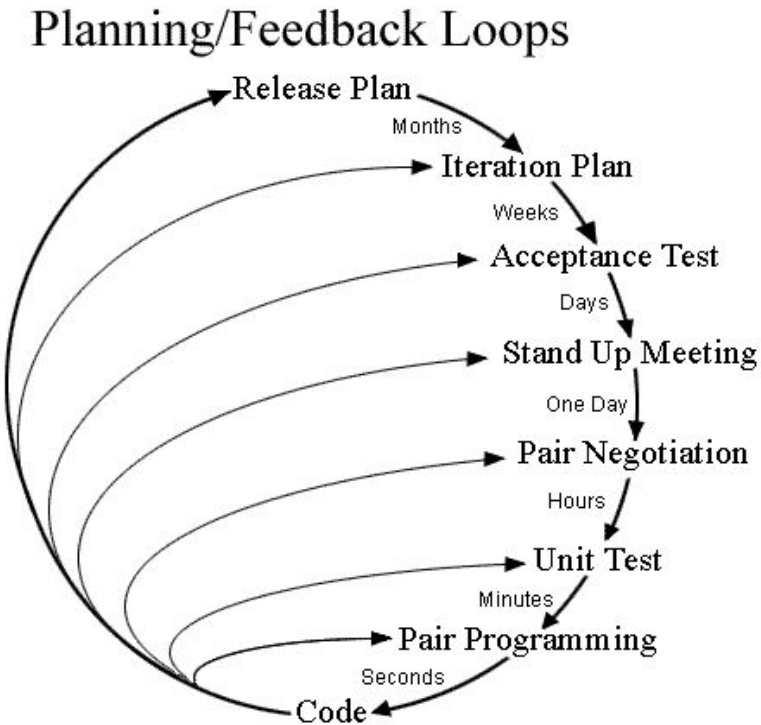
- авралы и переработки вредны на перспективе
  - работоспособность падает
  - “выгорание”
- жертвенность на работе -- признак непрофессионализма


On-site Customer
Planning Game
40 Hour Week
Metaphor
Refactoring
Simple Design
Short Releases
Testing
Pair Programming
Coding Standards
Collective Ownership
Continuous Integration

Процесс XP


Exploration
Iteration Planning
Iteration
Customer Approval
Small Releases
Stories
Stories
Pair Programming
Development
Testing
Acceptance Tests
Release
Estimates
Continuous Integration
System Metaphor
Architectural Spikes
Tests
Collective Codebase
Spikes

Итерации в XP



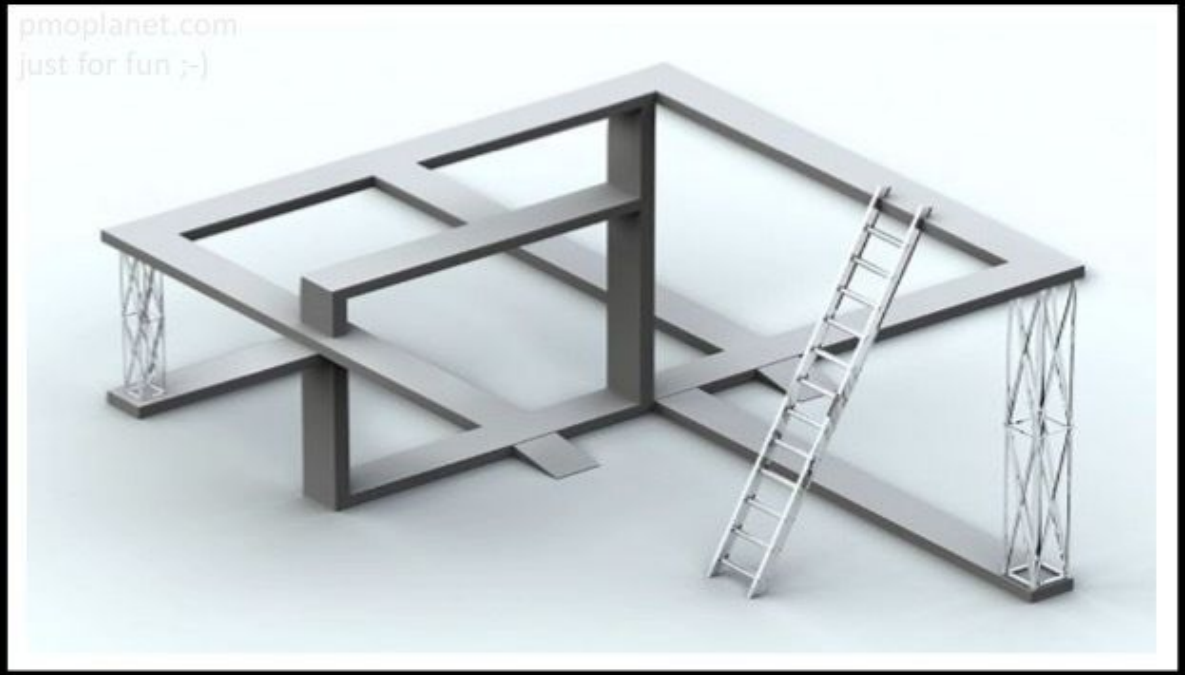
XP: резюме

- 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





AGILE DELIVERY

It's perfect for when the customer doesn't know what they want, right?