Quality Control in Continuous Integration

Konstantin Zhukov

Программа

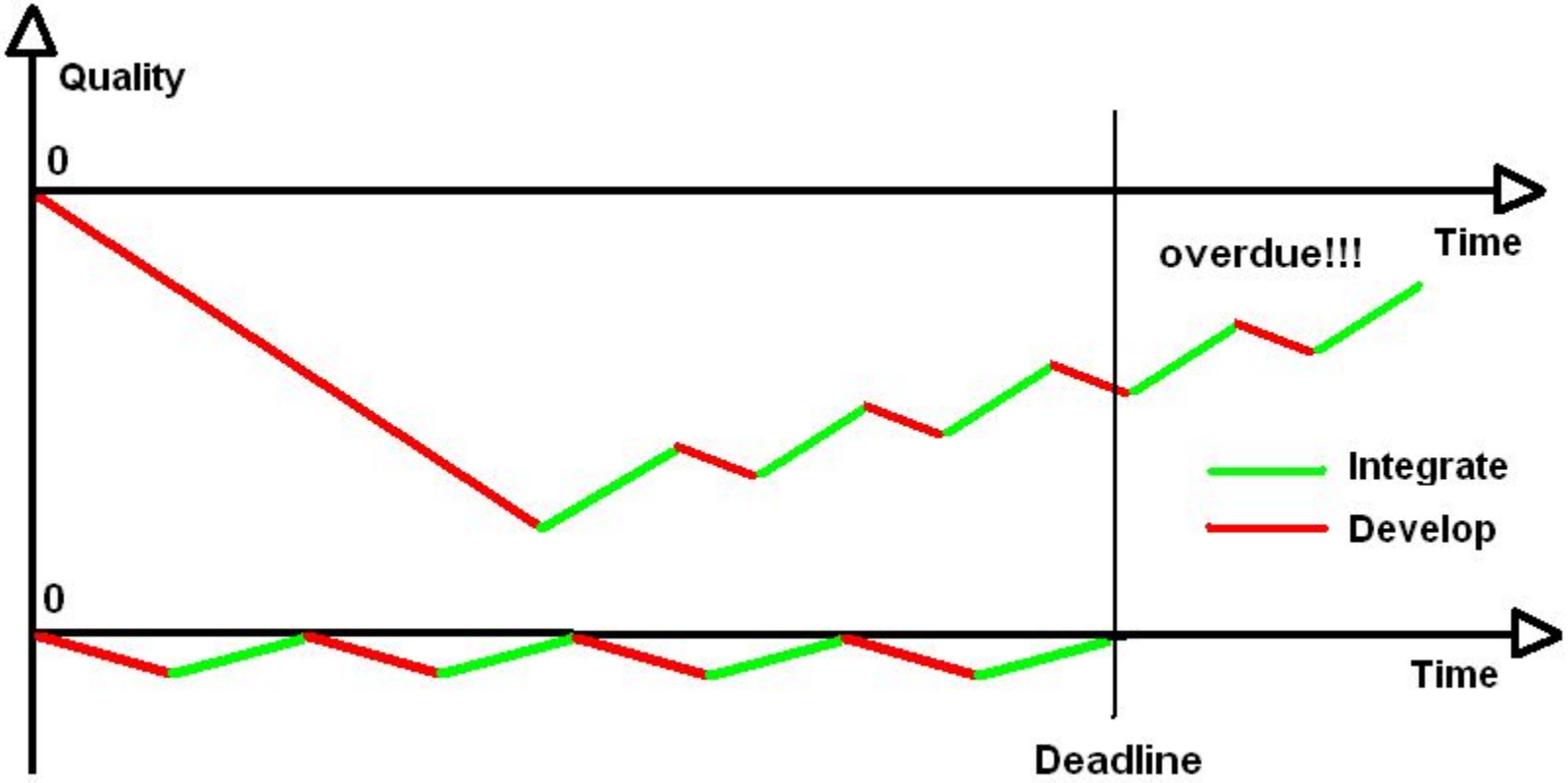
- Что такое Continuous Integration?
- Риски процесса разработки
- От «Continuous Integration» к «Build Pipeline»
  - *Quality Control*
  - *Практические моменты реализации*

Что такое Continuous Integration?

- «Process of applying quality control during development» (c)wikipedia
  - *Стратегия разработки,*
  - *связанная с регулярной интеграцией,*
  - *проводимой в автоматическом режиме*

Производство программных продуктов – рискованное дело

Риск 1: Поздняя интеграция стоит дорого



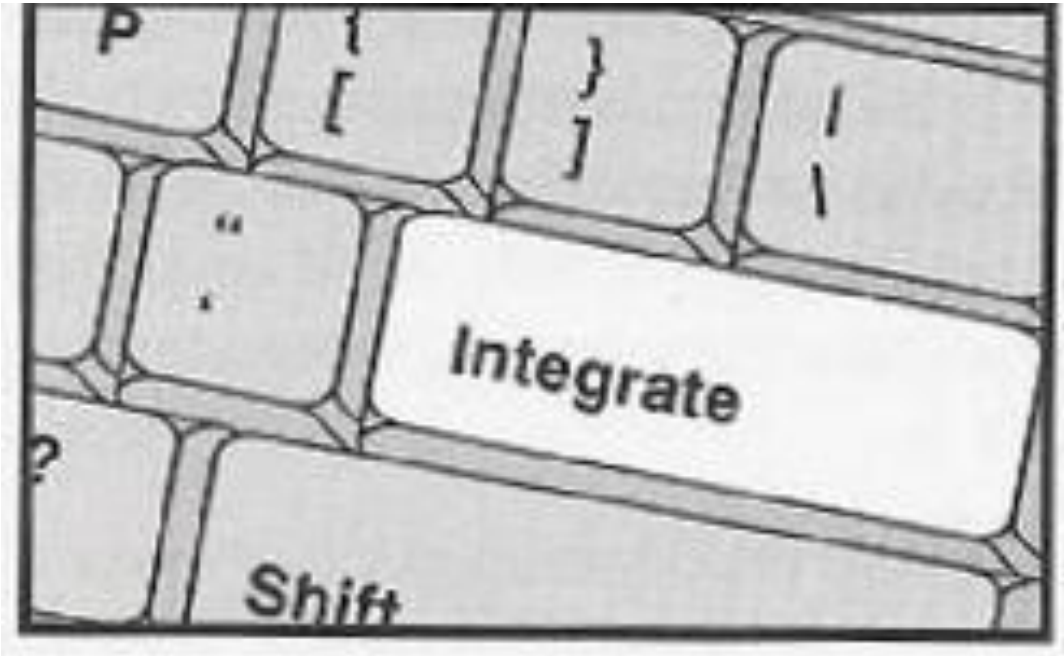
- *Атомарные изменения*
- *Интегрируемся чаще!*

Риск 2: Отсутствие регулярных сборок

- Продукт работает только локально
  - *Локально всё работает!*
  - *Завтра приедет босс, показывать нечего!*
- Процесс непрозрачен
  - *Какой сейчас статус проекта?*
  - *Что мы такого сделали в версии 1.0.1?*

Интеграция необходима

- Хотелось бы иметь что-то вроде

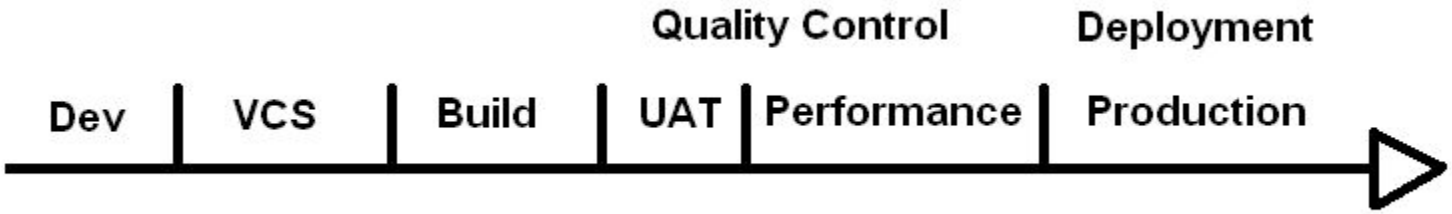


- Что дальше?

От «Continuous Integration» к «Build Pipeline»

- «Automated manifestation of your process for getting software from version control into the hands of your users»
- Continuous Integration flow

- Build Pipeline flow



Build Pipeline: Как это организовать?

- Специальные инструменты для поддержки процесса

ThoughtWorks

anthillpro
by urban{code}

- *… сегодня про это не говорим*

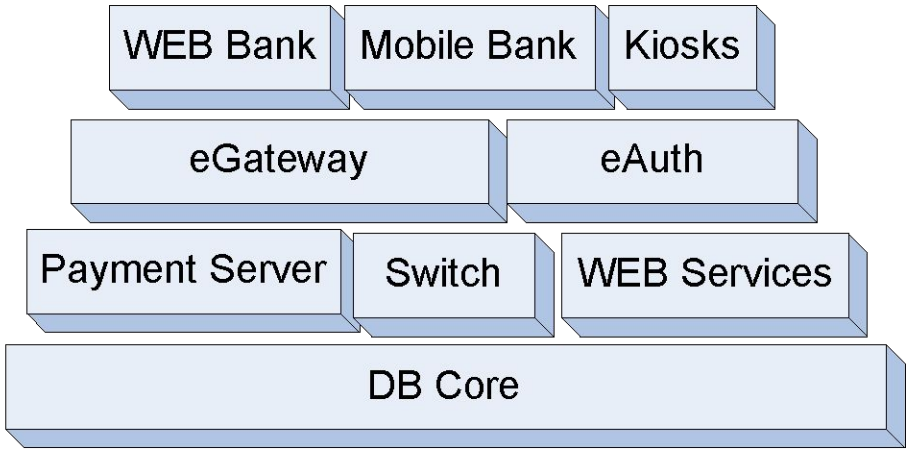
Build Pipeline: Quality Control

- Сконцентрируемся на QC
- Какие для этого предпосылки?
  - *К фазе QC доступны все необходимые артефакты (binaries)*
  - *Билд готов к тестированию!*
- Как его организовать?
  - *Есть проблемы*
  - *и есть решения*

Проблема 1: Слишком много продуктов

- Много продуктов -> слишком большая энтропия
- Все продукты разные -> разные инструменты
- Нужны:
  - *Правила организации тестов*
  - *Единая система управления разнородными тестами*

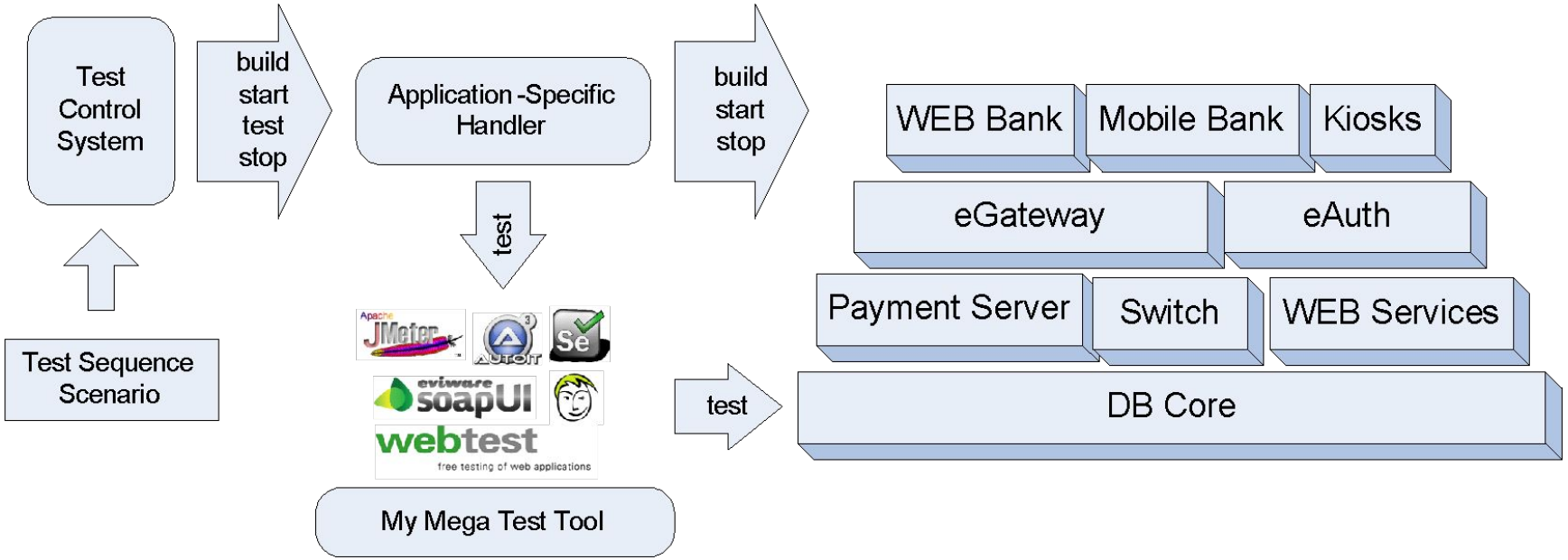
VS



Решение 1: Слишком много продуктов

- Основные шаги – общие для всех
- Идём от алгоритма
- Что нужно?
  - *Собрать конфигурацию приложения (build)*
  - *Запустить конфигурацию (deploy + start)*
  - *Запустить тесты (test)*
  - *Собрать отчёт (collect logs)*
  - *Остановить конфигурацию (stop + undeploy)*
- Отделяем управление от реализации

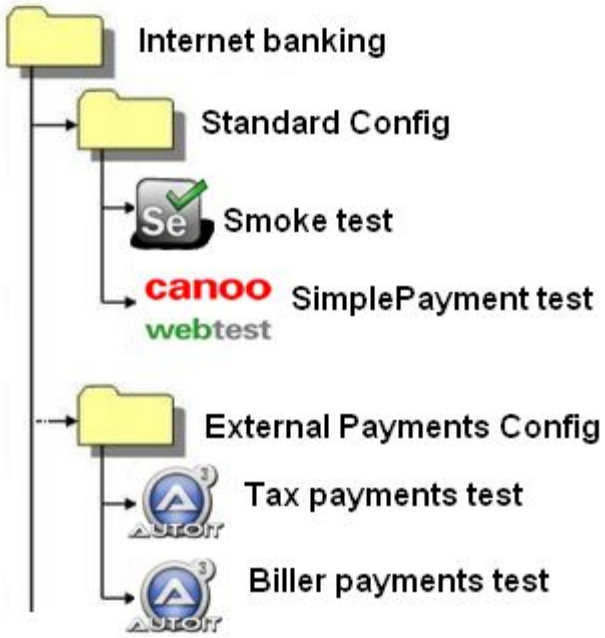
Система управления тестами: Алгоритм



- *Вся специфика тестируемых приложений спрятана в специальном handler-е*

Система управления тестами: Последовательность тестов

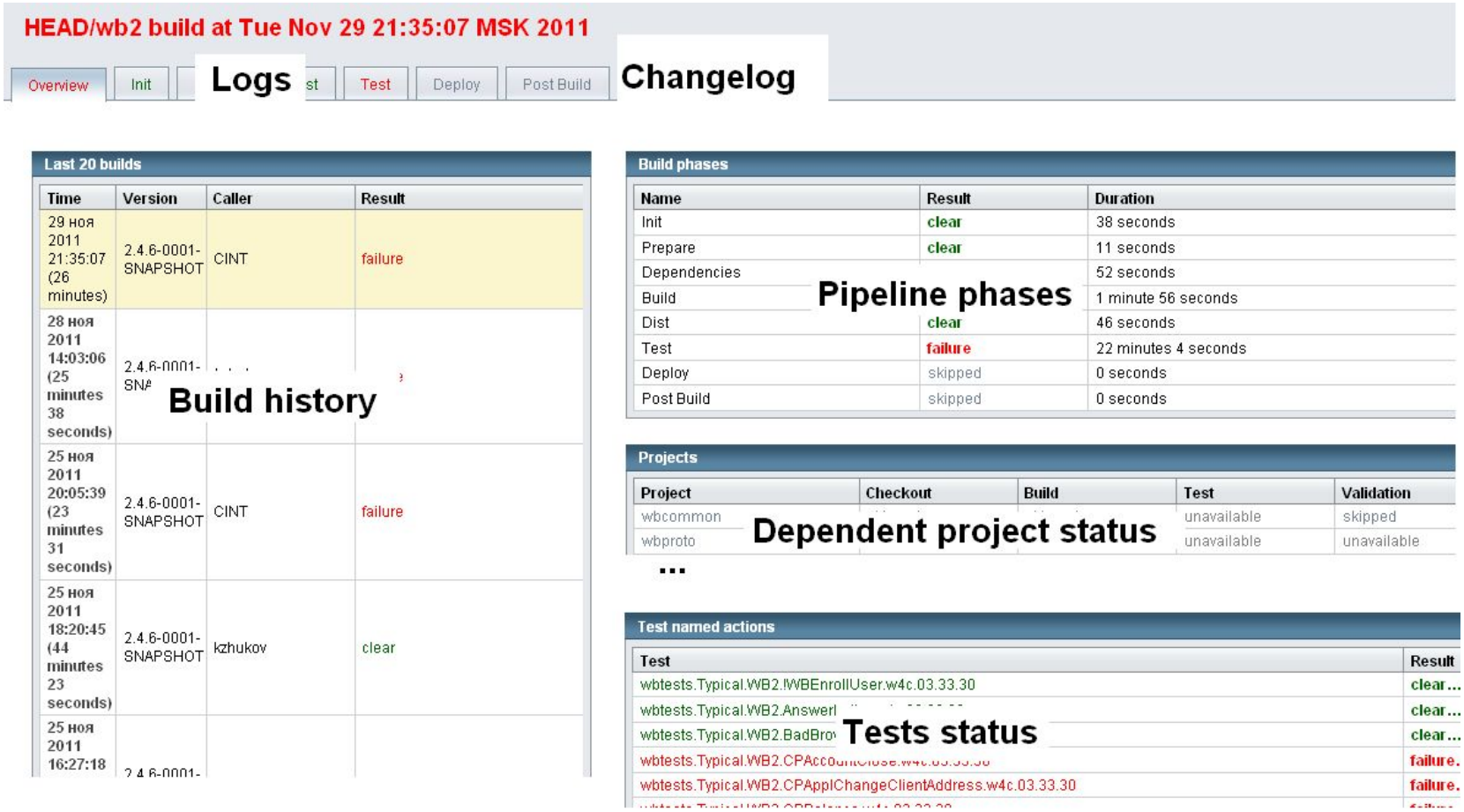
- Основа – файловое дерево
  - *Алгоритм обхода – итерирование на одном уровне*



  - *Простота!*

Система управления тестами: Мониторинг

- Единая WEB консоль CI сервера



Build Pipeline: Бесплатное приложение

- Ручной redeploy дорог
  - *Можно ли его автоматизировать?*
- Можно
  - *Меняем местами шаги start <-> stop*
  - *Создаём технический сценарий без тестов*
- В итоге:
  - *Приложение всегда up-to-date и готово к тестированию*
  - *… и к показу боссу ☺*

Проблема 2: Продукт слишком сложен

- Продукт слишком сложный, но нужны интеграционные тесты
- Разбираем его на части, сохраняя общую структуру,
  - *Собираем каждый компонент в режиме redeploy*
  - *Запускаем интеграционные тесты из общей точки входа*

Итоги

- Что мы добились
  - *Разрешение всех рисков*
  - *Покрытие всех продуктов компании*
  - *+ Помощь в ручном тестировании*
- С небольшими недостатками
  - *Поддержка непростой инфраструктуры*
  - *Необходимость отдельных серверов*
  - *Зачем тестировать самому, если CI всё проверит сам? ☺*

Thanks

Thank you!

kzhukov@ecelon.com
kost.zhukov@gmail.com

SQA DAYS#10
Точка роста твоей карьеры