Проектирование ПО

Лекция 16: Case Studies of Open-Source Systems

Тимофей Брыксин
timofey.bryksin@gmail.com

CMake

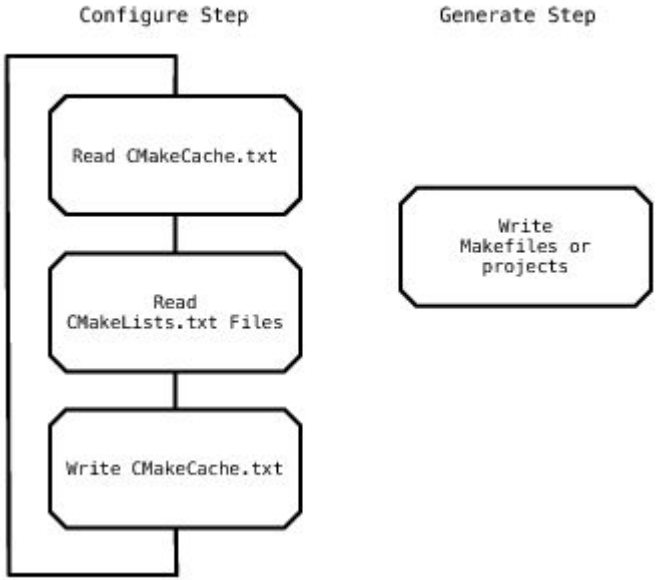
- 1999 год
  - Makefile для *nix, проектные файлы MSVS для Windows
- инструмент для конфигурирования, сборки и развёртывания ПО на разных платформах
- замена autoconf/libtool

CMake: требования

- зависимость только от компилятора C++
- генерация проектных файлов Visual Studio
- работа с целями сборки
  - статические и динамические библиотеки, запускаемые файлы
- возможность создания кодогенераторов времени сборки
- различные варианты дерева сборки по одному дереву кода
- автоматическое сканирование зависимостей заголовочных файлов
- кроссплатформенность

CMake: процесс

- фаза configure
  - построение промежуточного представления процесса сборки
  - CMakeCache.txt
  - CMakeLists.txt
  - паттерн “Команда”
- фаза generate
  - создание файлов для сборки под целевое окружение



CMake: архитектура

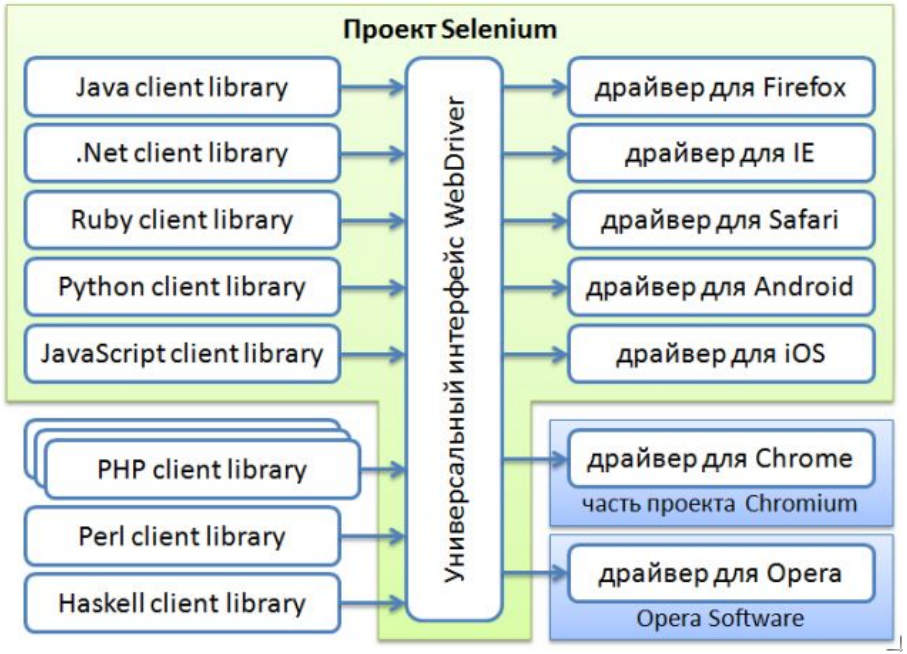
- object-oriented system using inheritance, design patterns and encapsulation
- управление зависимостями
  - depend.make
  - flags.make
  - build.make
  - DependInfo.cmake

CMake: lessons learned

- agile процесс разработки
- обратная совместимость важна
- собственный встроенный язык -- плохая идея
- плагины не выстрелили
- не стоит раскрывать много API

Selenium

- средство для автоматизации GUI-тестов для веб-приложений
- Selenium IDE
  - record/playback paradigm
- WebDriver
- Selenium Grid



Selenium: история

- Selenium Remote Control
  - система манипулирования HTML на JS
  - работа тестов на сервере вместе с приложением в одной JS песочнице
  - HTTP прокси для внешнего взаимодействия
  - API для разных языков
- WebDriver
  - взаимодействие с браузером со стороны
  - изоляция внешнего взаимодействия через паттерн Адаптер
  - объектно-ориентированный API для Java

Selenium: architectural themes

- эмуляция действий пользователя
  - события браузера и события ОС
- управление сложностью
  - спрятать сложность от пользователя
  - понижение bus factor’a
- поддержка реализаций JS
- большое количество удалённых вызовов
  - задержки
  - гранулярность API
- открытый проект

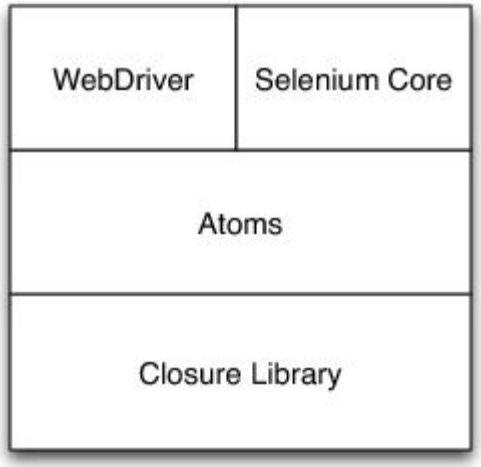
WebDriver API

- object-based API, role-based интерфейсы
  - сложность изучения
- X браузеров, Y языков
  - унификация API разных браузеров
  - перенос логики в драйверы (например, запуск браузера)

```
public void userWrongAuth() {
       $(byText("Password or login wrong")).shouldNotBe(exist);
       char[] alphabet = "abcdefghijklmnopqrstuvwxyz0123456789".toCharArray();
       String wrongLogin = RandomStringUtils.random(20, alphabet);
       String wrongPassword = RandomStringUtils.random(20, alphabet);
       $(By.name("username")).setValue(wrongLogin);
       $(By.name("password")).setValue(wrongPassword);
       $("[type=\"submit\"]").click();
       $(byText("Password or login wrong")).shouldBe(exist);
   }
```

Уровни и JavaScript

- средства манипуляции DOM
  - Closure Library + Closure Compiler
    - примитивы, модуляризация
    - компиляция, оптимизация и/или генерация в C-style константу
  - атомы как минимальные единицы автоматизации
    - манипуляция свойствами элементов, события
    - переиспользование кода
    - отличия в реализации браузеров
- средства выполнения JS
- средства эмуляции пользовательских событий

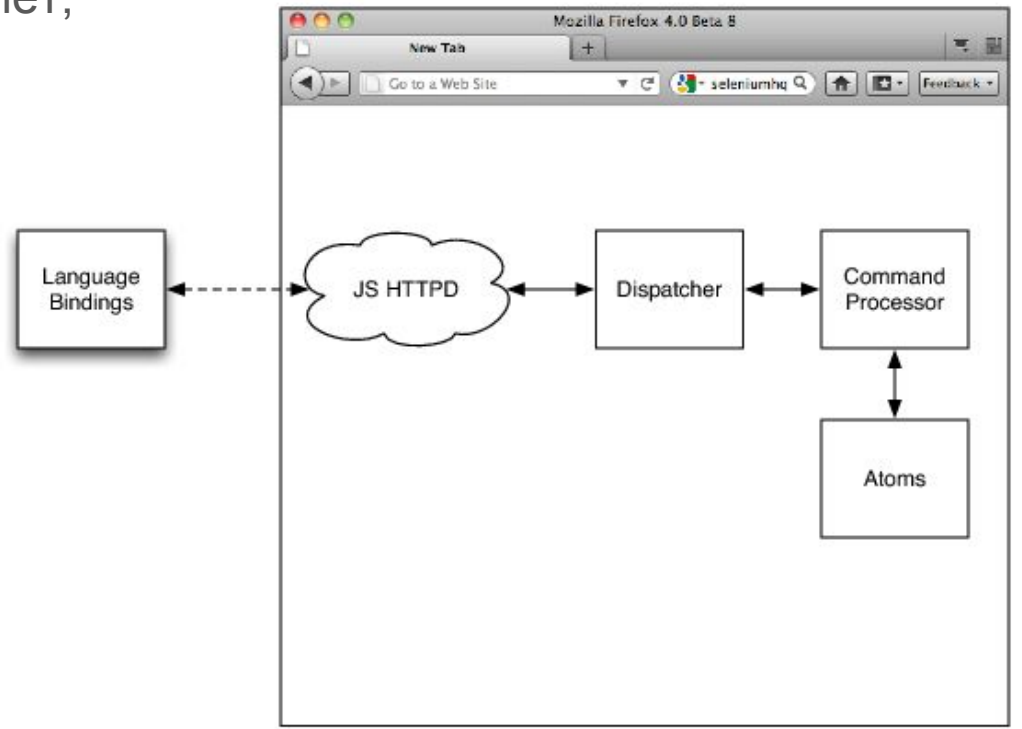


Удалённый драйвер/драйвер Firefox

- WebDriver -- единый интерфейс для всех браузеров
- v1
  - Mozilla XPCOM фреймворк
  - активное использование RPC
    - самописный текстовый line-oriented протокол
      - самописные библиотеки
      - не поддерживает бинарные файлы
- v2
  - HTTP + JSON
    - single end-point vs REST
    - коды возврата 4xx и 500
  - общие среди всех драйверов статус-коды операций

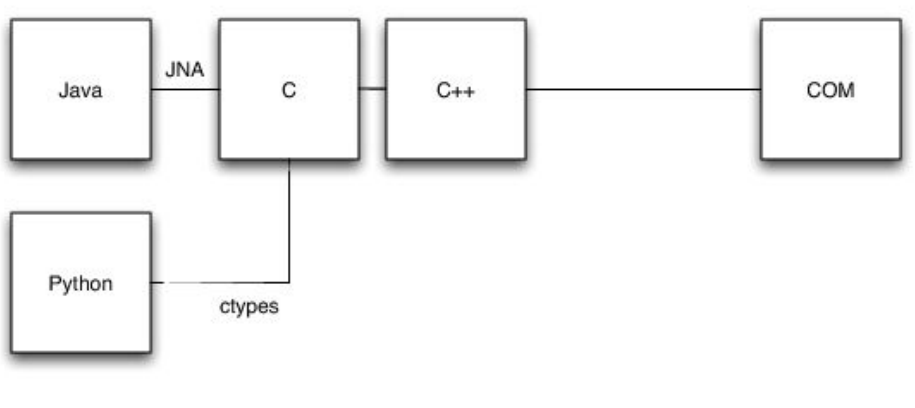
Драйвер Firefox (2)

- WebDriver Server как Java-сервлет, перенаправляющий команды нужному экземпляру WebDriver
- Firefox драйвер реализован как Firefox extension



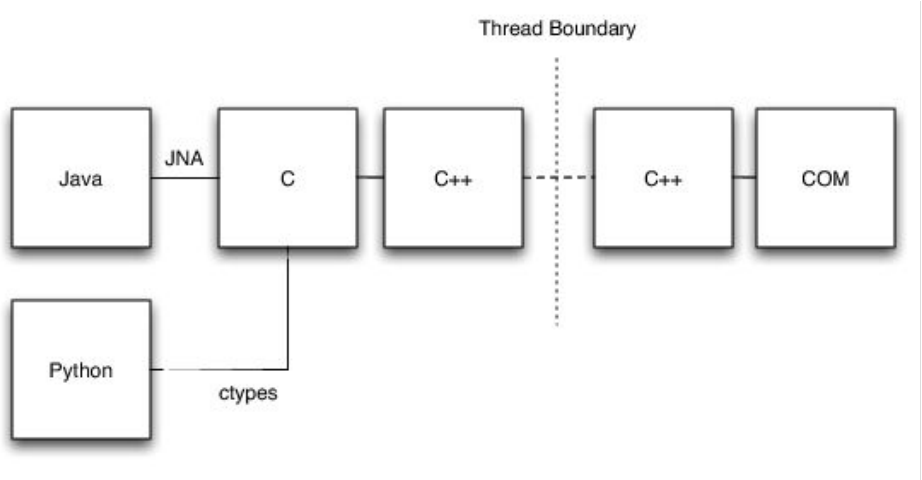
Драйвер IE

- IE как набор COM компонент
  - обратная совместимость к началу времён
- архитектурная особенность -- отсутствие инсталлятора
  - сложно, долго, непривычно
  - C++, статическая линковка со стандартными библиотеками
- промежуточный C-интерфейс



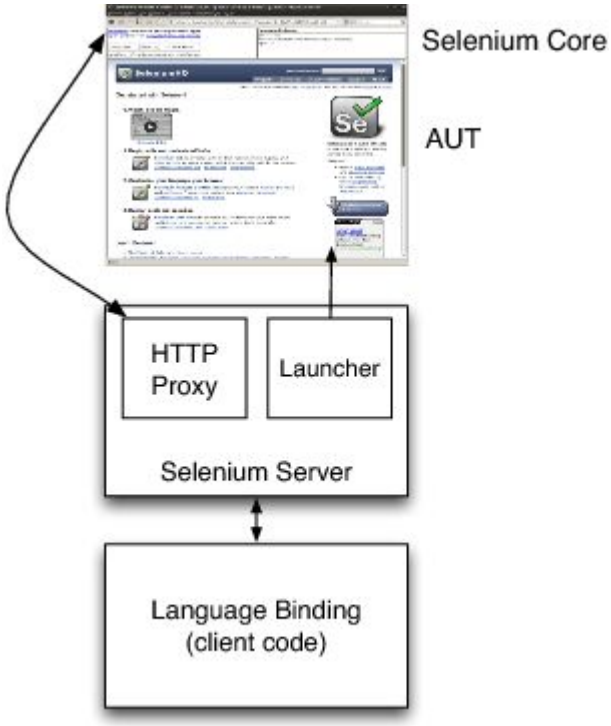
Драйвер IE: многопоточность

- однопоточная модель COM Automation интерфейса
  - однопоточный executor и доступ через Future на сервере приложений
  - отдельный поток для экземпляра IE и взаимодействие через PostThreadMessage
    - таймаут для обнаружения аварий
    - сложность отладки драйвера
      - высокий bus factor
- переход на Automation Atoms
  - компиляция JS кода в C++ константы



Selenium RC

- когда не удаётся напрямую работать с браузером
- Selenium Core -- фреймворк на чистом JS
- сервер как прокси к браузеру
- асинхронный Response/Request
  - попытка побороть Single Host Origin Policy
  - альтернативы -- long polling и busy waiting



Selenium: выводы

- сложность интеграции возможностей разных браузеров
  - It's all edge cases!
- уровни абстракции -- это хорошо
- прятать от пользователей сложность в драйвер -- это хорошо
- драйвера браузеров -- это и хорошо, и плохо
  - дают больше возможностей
  - очень сложны и должны быстро эволюционировать

Mercurial

- python + C
- распределённая VCS
- требования
  - миллионы файлов
  - миллионы ревизий
  - тысячи пользователей, вносящих изменения параллельно
    - десятилетиями
- challenges
  - компрессия хранилища данных
  - эффективная работа с произвольными ревизиями
  - эффективное добавление новых ревизий
  - работа с историями файлов

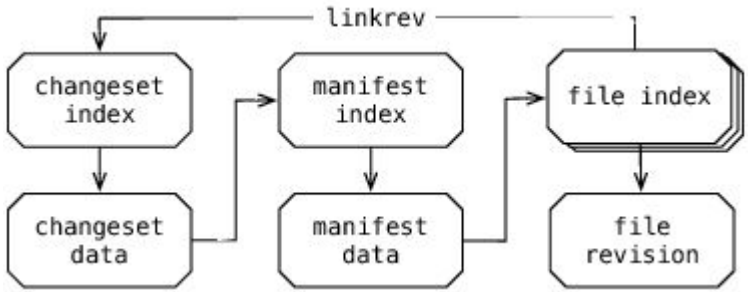
Revlog

- способ хранения информации о файле
- файл индекса
- файл данных
  - дельты и полные версии файлов
  - zlib сжатие
    - если необходимо

6 bytes	hunk offset
2 bytes	flags
4 bytes	hunk length
4 bytes	uncompressed length
4 bytes	base revision
4 bytes	link revision
4 bytes	parent 1 revision
4 bytes	parent 2 revision
32 bytes	hash

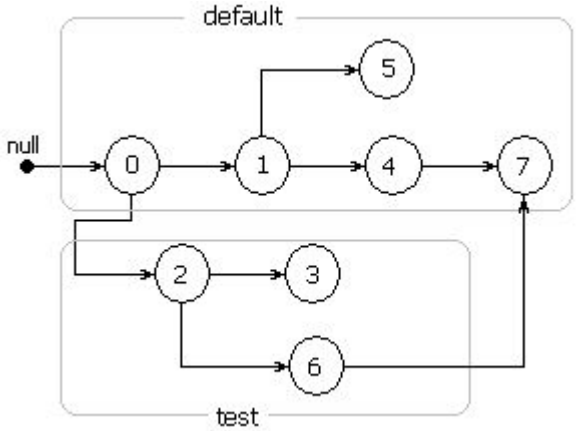
Модель данных

- changelog
  - метаданные о ревизии (дата, имя, ...) + ссылка на манифест
- manifests
  - список имён файлов в ревизии + для каждого ссылка на filelog
- filelog
  - содержимое файлов ревизии + немного метаданных
- dirstate
  - текущая ревизия
  - кэширование информации о дереве файлов
- последовательное обратное обновление логов
  - дописывание в конец
  - транзакции



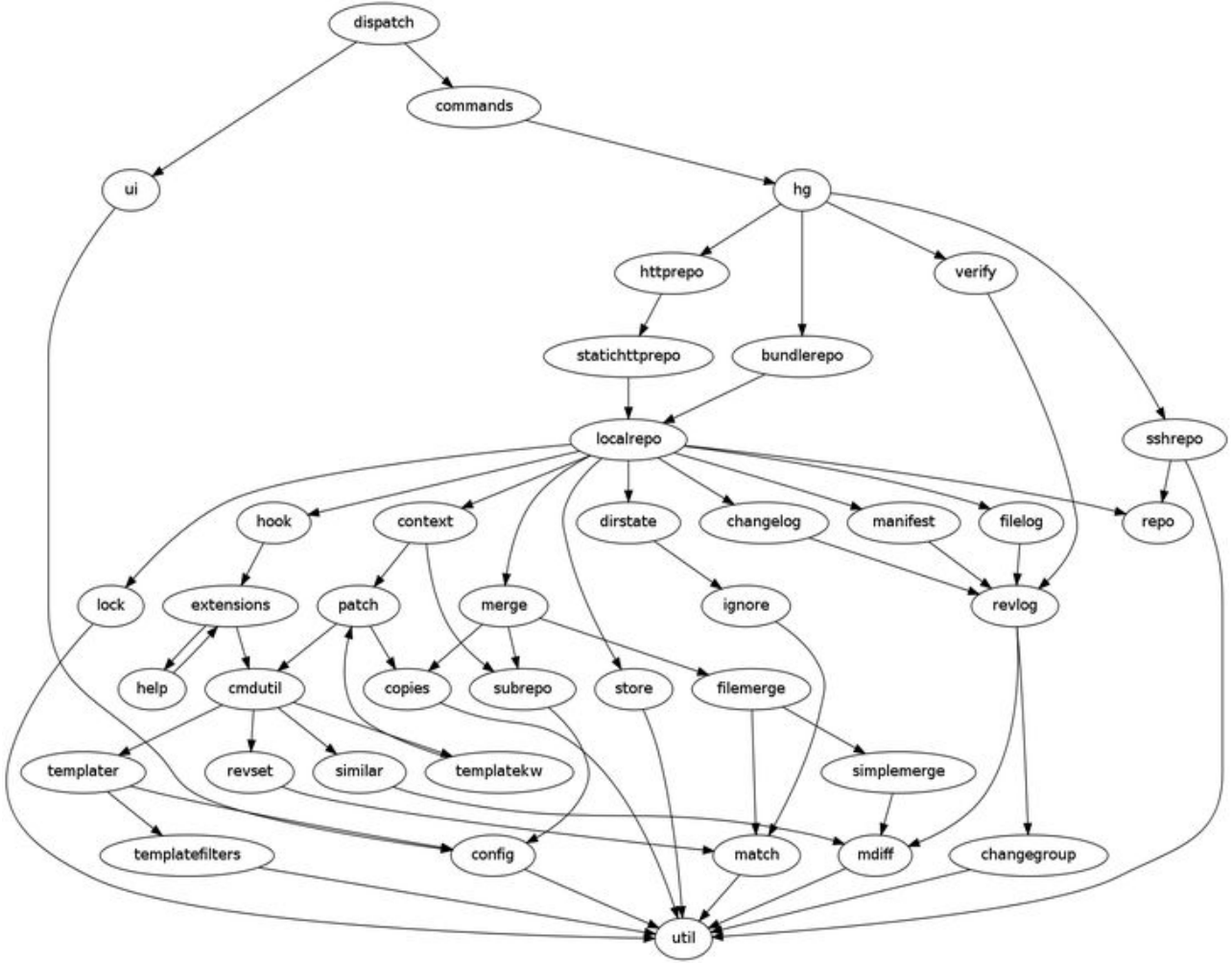
Механика версионирования

- создание ветки через клонирование репозитория
  - суровый хардкор
- bookmarks
  - объекты-ссылки аля git refs
- именованные ветки
  - поле с именем в метаданных changeset’a
- анонимные ветки
  - ветки и головы



Архитектура

- CLI
  - вывод
  - запрос аргументов
  - вызов стороннего приложения



Расширяемость

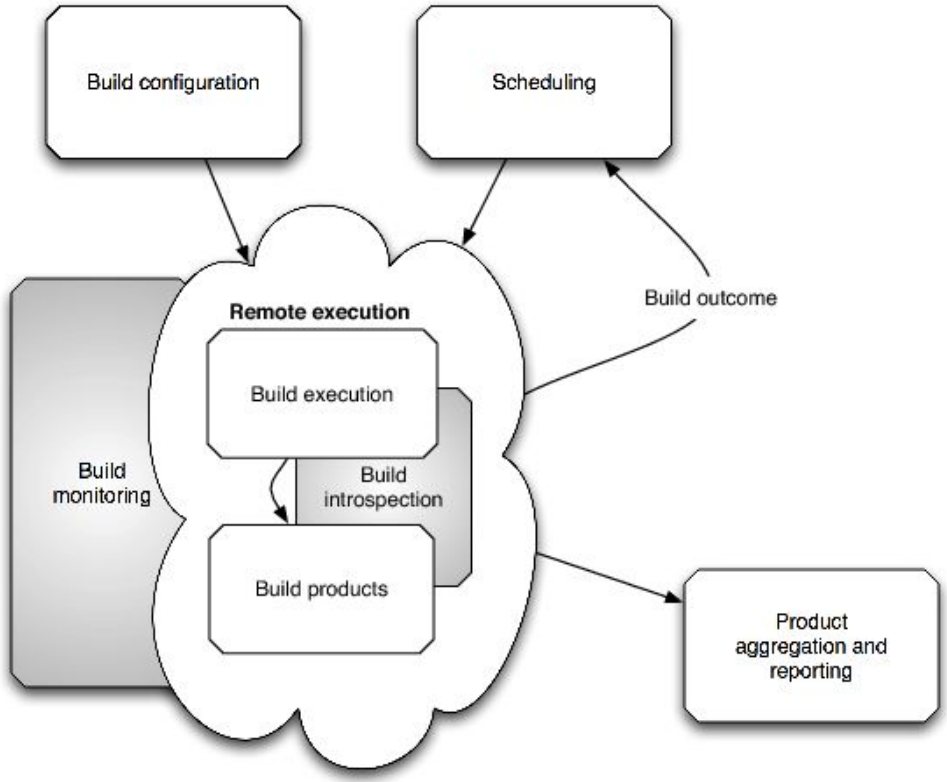
- модули расширения
  - новые команды
  - обёртки над существующими командами
  - обёртки над репозиторием
  - обёртки над любой функцией Mercurial
  - новые типы репозиториев
- hooks
  - вызов шелл-скрипта
  - вызов Python-функции

Mercurial: выводы

- Python: и хорошо, и плохо
- сложно модифицировать changeset после публикации
- revlogs + модель данных -- хорошо и эффективно
- небольшое количество основных команд -- хорошо
- разные виды сжатия -- хорошо

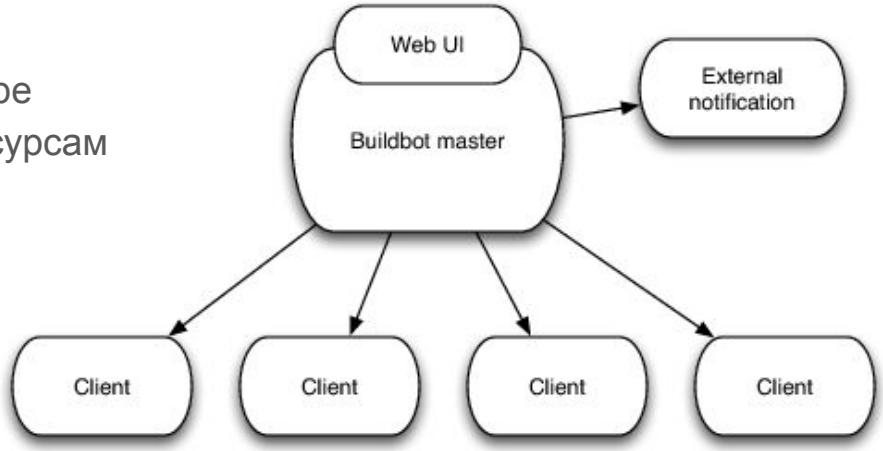
Continuous Integration Systems

- build, test and report
  - “чистая” система
  - разные платформы и окружения
    - рецепты сборки
  - сбор статистики
  - сборка релизов
  - внешние инструменты
- основные подходы к реализации
  - master/slave
  - reporting architecture
  - гибридные модели



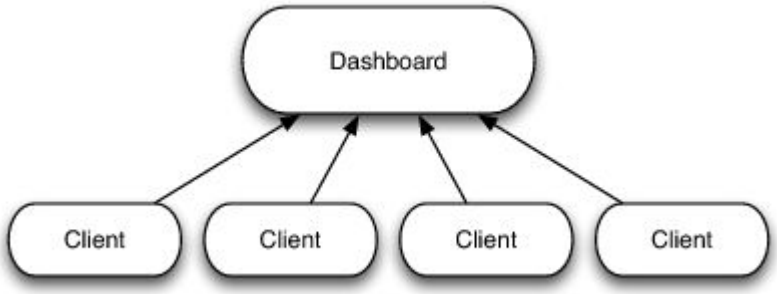
Buildbot: архитектура master/slave

- центральный сервер и ряд подчинёнnных клиентов
- планирование команд и контроль выполнения мастером в реальном времени
  - постоянное соединение с клиентами
  - выполнение команд по одной и возврат результатов
    - конфигурирование сборки на сервере
  - координация конкурентного доступа к ресурсам
  - фиксированный набор клиентов
- отсутствие механизмов контроля ресурсов клиентов



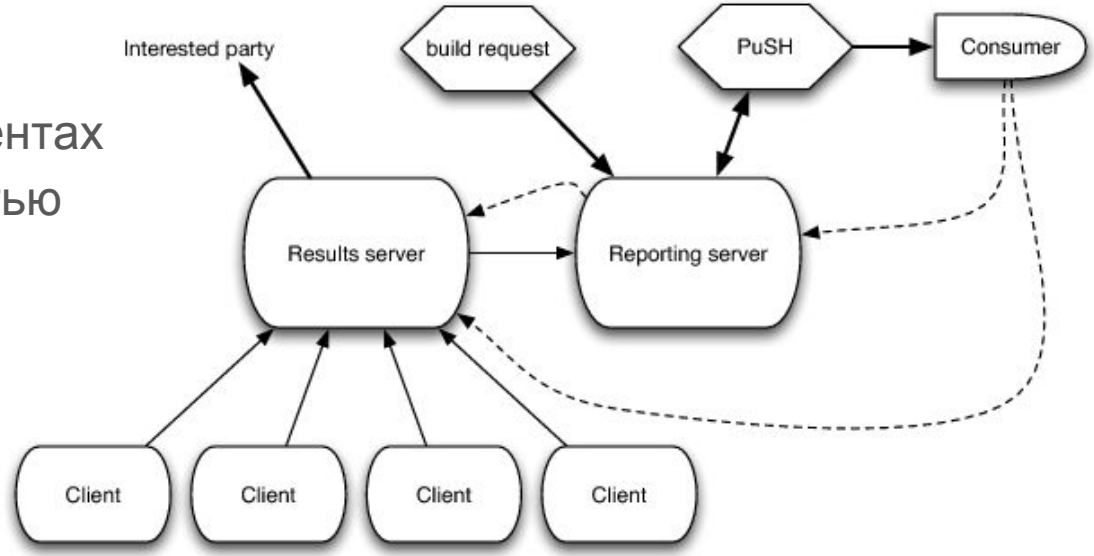
CDash: reporting server model

- сервер для сбора информации с узлов, запускающих CMake и CTest
- хранилище результатов работы автономных клиентов
  - все решения принимают клиенты
  - распределённая система с динамическим конфигурированием
- нет централизованного управления процессом
  - не отследить прогресс
  - невозможен глобальный запуск сборки
    - необходим polling
- страдает от ненадёжных клиентов



Pony-Build: децентрализованная архитектура

- loosely coupled RPC and webhook callback-based model
- координирование сборки на клиентах
- разграничение ответственностей
- гибкость настройки
- простота коммуникаций
- конфигурирование на клиентах
- проблемы с конкурентностью
- проблемы с real-time мониторингом процессов



Jenkins: гибридная модель

- три в одном
  - standalone CI система
  - координатор удалённых сборок
  - пассивный сборщик информации
- pipeline flow
- разные механизмы запуска задач
- плагинная система для расширения
- централизованная блокировка ресурсов

Выводы

- функциональность определяет архитектуру
- орг. структура определяет архитектуру
  - вариант закона Конвея
- распределённость vs контроль