Introducing

Calaba.sh

automated functional testing of mobile apps

Karl Krukow,
CTO, LessPainful
GotoAMS, May, 2012
karl@lesspainful.com, @karlkrukow

About me

- PhD Computer Science, University of Aarhus, 2006
- Developer at Trifork for about 6 years
  - Java enterprise, web tech, JavaScript
  - Ruby, Clojure
  - Last two years as iOS developer
- Present: CTO & iOS responsible at LessPainful

Agenda

- Automated functional testing for mobile
  - Some desirable properties for a functional testing tool
- Introduce Calabash
  - Focus on iOS only
- Live Demo:
  - Calabash iOS
  - LessPainful: test service and device cloud

Professional practices?

- Invest the time :tices going.
  Examples:
  - Unit testing
  - Functional tests
  - Continuous buil
  - Continuous unit e coverage).
  - Static analysis us
  - Automated depl , p updates or link in email.
  - Automatic Crash/Error reporting.
  - ...

The functional testing story for mobile apps

- Many devices, screens, OS versions, languages.
- Often a manual process: repetitive, expensive.
  - Regressions, e.g., app crashes
- Visual appearance of screens matter (alot!)
  - User experience, Design guidelines, branding,...
- As realistic an environment as practically possible.
  - simulators/emulators are good, but not enough!

Automated functional testing desiderata

- Minimize distance between use cases and actual test code (DSLs?).
- Expressive and efficient to write.
- Extensible
- High-level, declarative (robustness against “minor” UI changes).
- Support testing in realistic environments (multiple real devices, on multiple OS versions, languages).
- Support Continuous integration.


Cucumber

- Cucumber provides
  - a notation for writing software specifications
  - a software tool for executing those specifications
- Specifications are written in a business readable language that is close to natural language.
- Extremely popular tool for test and specs of web applications.
- http://cukes.info/

Cucumber Example

```
Feature:  As an administrator. I want to be able to add and remove users,
          so I can control access to the application

Scenario: Add test user
    When I touch the Add User button
    And I fill in text fields as follows:
      | field      | text   |
      | Last Name  | Knorr  |
      | Username   | knorr  |
    And I touch "Save"
    Then I should be on the Users screen
    And I should see a table containing "Knorr"

Scenario: ...
```

Step Definitions

- Make the cucumber tests “come alive”
- Written in ordinary programming languages
  - Mostly Ruby (but cucumber-jvm: Java, Clojure,...)

Feature

```
Scenario: Add test user
    When I touch the Add User button
...
```

Step definitions

```
When /^I touch the Add User button$/ do
  btn_txt = 'Add user'
  touch("button text:#{btn_txt}")
end
```

Execution

- Executing a test produces a test report
  - for each step, did it succeed or not
    - exception/error message if present
- Test report formats
  - Machine readable (XML, JSON,...)
  - Human readable, (HTML, console)
  - your own...

Calaba.sh

Calabash

- *One* interface: ***Cucumber***, for *Android and iOS*.
  - Predefined and custom steps: APIs in Ruby + JVM(wip)
  - Reuse of Cucumber features across platforms possible.
  - Runs on physical devices and simulators.
  - Support for hybrid apps (embedded webviews)
  - Free, open source *with optional commercial extras support, training, consulting*, device cloud, *private* device cloud, enterprise cloud...

LessPainful
Test Execution Service

- Execute Calabash tests concurrently on many devices, OS'es, languages.
- Visual test reports.
  - Comparison across models and operating systems.
- Authentic: Not jailbroken, iOS and Android devices, rotation.
- Continuous integration:
  `calabash-ios submit app.ipa KEY`

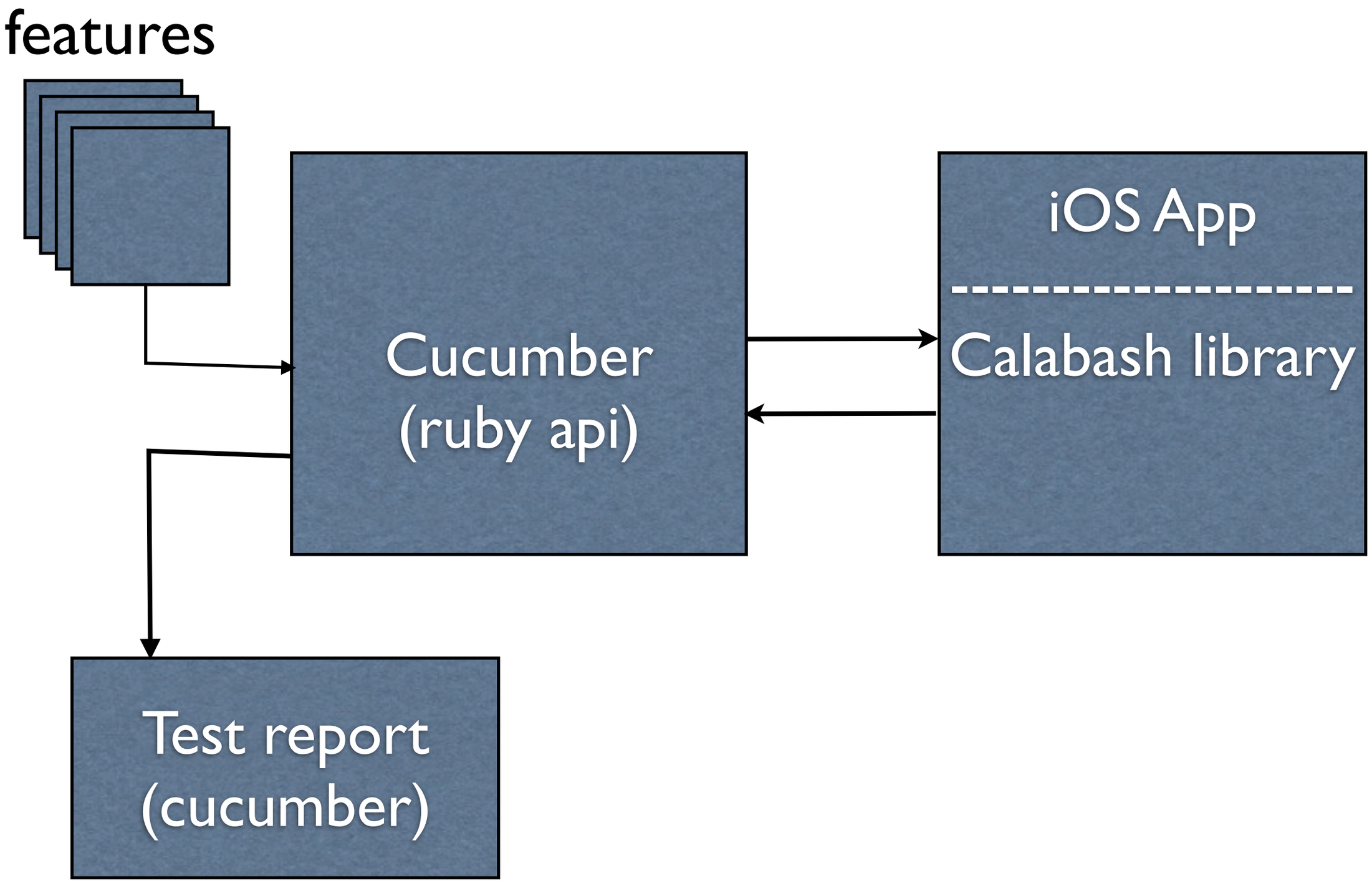
Mobile Test Lab

www.lesspainful.com

Calabash iOS:

more detail

Architecture iOS



Calabash iOS

- Very easy to get started for iOS developers/QAs.
- Declarative query language for finding views.
  - Based on UISpec, but *simplified and extended*. (New Implementation, EPL licensed).
- Advanced touch synthesis.
  - Supports gestures (pan, swipe, pinch, multitouch,...)
  - Extensible.
- Full power of Ruby programming language for test logic
  - Supports interactive, exploratory test development.
- Can use device accessibility for identifying views.

Queries

- Queries are like CSS selectors or XPath
  - `label text:'Hello'`
  - `label index:2`
  - `view marked:'thepane' label`
  - `view:'MyClassName'`
  - `label {text LIKE 'Hel*'}`
  - `webView css:'#header a.cssclass'`
  - `webView xpath:'//node()'`

Demo:

- Calabash iOS

- LessPainful Device Cloud

iOS Comparisons

- Several options available. To my knowledge:
  - Calabash
  - UIAutomation, Apple
  - Zucchini, iOS Testing Framework
  - Frank, Pete Hodgson, ThoughtWorks
  - UISpec, http://code.google.com/p/uispec/
  - FoneMonkey => MonkeyTalk, GorillaLogic
  - KIF, Square
  - NativeDriver, http://code.google.com/p/nativedriver/

References

- https://github.com/calabash
  - https://github.com/calabash/calabash-ios
  - https://github.com/calabash/calabash-ios/wiki
    - https://github.com/calabash/calabash-ios-server
    - https://github.com/calabash/calabash-android
- http://blog.lesspainful.com/
- https://www.lesspainful.com/

Questions?

Making app testing less painful...
Please contact us with any questions:
contact@lesspainful.com
karl@lesspainful.com - iOS
jonas@lesspainful.com - Android

http://www.lesspainful.com