Automated Testing of Mobile Apps
Automated Testing of Mobile Apps

Karl Krukow
Xamarin
Tech Lead, Xamarin Test Cloud
@karlkrukow
<table>
<thead>
<tr>
<th>Zone</th>
<th>Turnumre</th>
<th>Hverdag ord.</th>
<th>Hverdag undt.</th>
<th>Hverdag lørdage</th>
<th>Hverdag undt. lørdage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Busselskab</td>
<td></td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>Rute 107 fra Horsens an</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>06.40</td>
</tr>
<tr>
<td>340 Skanderborg Bustransport af</td>
<td>04.40</td>
<td>05.15</td>
<td>...</td>
<td>...</td>
<td>05.45</td>
</tr>
<tr>
<td>340 Skanderborg Station</td>
<td>04.44</td>
<td>05.19</td>
<td>...</td>
<td>...</td>
<td>05.49</td>
</tr>
<tr>
<td>340 Stilling</td>
<td></td>
<td>05.26</td>
<td>...</td>
<td>...</td>
<td>06.15</td>
</tr>
<tr>
<td>342 Hørning</td>
<td></td>
<td>05.31</td>
<td>...</td>
<td>...</td>
<td>06.32</td>
</tr>
<tr>
<td>307 Hasselager</td>
<td></td>
<td>05.38</td>
<td>...</td>
<td>...</td>
<td>06.39</td>
</tr>
<tr>
<td>303 Viby Torv</td>
<td></td>
<td>05.45</td>
<td>...</td>
<td>...</td>
<td>06.47</td>
</tr>
<tr>
<td>301 Banegårdspladsen</td>
<td>05.19</td>
<td>05.54</td>
<td>...</td>
<td>...</td>
<td>06.57</td>
</tr>
<tr>
<td>301 Sønder Allé v. Rtb.</td>
<td>05.21</td>
<td>05.56</td>
<td>...</td>
<td>...</td>
<td>06.59</td>
</tr>
<tr>
<td>301 Dokk1</td>
<td></td>
<td>05.57</td>
<td>...</td>
<td>...</td>
<td>06.62</td>
</tr>
<tr>
<td>301 Nørreport</td>
<td></td>
<td>06.00</td>
<td>...</td>
<td>...</td>
<td>06.65</td>
</tr>
<tr>
<td>302 Randersvej/Skejbyvej</td>
<td>05.37</td>
<td>06.12</td>
<td>...</td>
<td>...</td>
<td>06.67</td>
</tr>
<tr>
<td>304 Lisbjerg Erhvervspark</td>
<td>05.40</td>
<td>06.15</td>
<td>...</td>
<td>...</td>
<td>06.70</td>
</tr>
<tr>
<td>325 Søften</td>
<td></td>
<td>06.20</td>
<td>...</td>
<td>...</td>
<td>06.75</td>
</tr>
<tr>
<td>325 Adalsvej/Herredsvej</td>
<td>05.51</td>
<td>06.26</td>
<td>...</td>
<td>...</td>
<td>06.81</td>
</tr>
<tr>
<td>325 Hinnerup C</td>
<td></td>
<td>06.30</td>
<td>06.42</td>
<td>07.12</td>
<td>07.33</td>
</tr>
<tr>
<td>Rute 115 mod Randers af</td>
<td>05.58</td>
<td>06.58</td>
<td>...</td>
<td>...</td>
<td>07.54</td>
</tr>
<tr>
<td>325 Rylevej/Nørregade</td>
<td>05.53</td>
<td>06.28</td>
<td>06.31</td>
<td>07.13</td>
<td>07.34</td>
</tr>
<tr>
<td>325 Nørregade/Kildevangen</td>
<td>05.55</td>
<td>06.45</td>
<td>...</td>
<td>...</td>
<td>07.51</td>
</tr>
<tr>
<td>325 Bjarregaardvej/Rylevej an</td>
<td>05.57</td>
<td>06.47</td>
<td>...</td>
<td>...</td>
<td>07.53</td>
</tr>
<tr>
<td>325 Rylevej/Bjarregaardvej</td>
<td>06.31</td>
<td>06.39</td>
<td>...</td>
<td>...</td>
<td>07.26</td>
</tr>
<tr>
<td>325 Foldby</td>
<td></td>
<td>06.51</td>
<td>...</td>
<td>...</td>
<td>07.38</td>
</tr>
<tr>
<td>328 Hammel rtb.</td>
<td></td>
<td>05.39</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

D Udgår fra Stillingbyvej/Gramvej 5.39 og 06.09.
C Kører af rute 109 til AUI, Skejby og Agro Food Park.
R Kører af rute 109 til AUI, Skejby og Agro Food Park.
Afgangstiden i Hørning gælder ved Torvet.
Crash to home screen!
High expectations
Mobile poses unique challenges

Market share

Number of devices

134

75%
Keep your hand up if...

“have automated our mobile app builds on a CI server”
“have unit tests and/or integration tests of our mobile app”
“automatically run UI tests on every commit or at least daily”
Keep your hand up if...

“develop specs/tests for all features as part of dev cycle”

“release regularly, flexibly, frequently and with confidence”

“rarely find bugs in manual testing”
Anyone still here?
It’s possible...

**eBay Denmark - case study**

- Reach: approx. 1/4 of the Danish population
- 4 iOS developers / (backend), 1 QA expert, 1 Scrum master
- Used to spend 3 days on manual regression testing before each release
  - Frequently found bugs during the release
- Stressful for the engineers and the manual tester
- Just wouldn't put up with it anymore
- Looked into automated testing.
Outcome?

Pain

Frustration
Outcome - the good parts!

Also Success :) 

Saw the potential benefits but ... 

is it worth it?
Why?

Tools immature, unsupported
Had to learn everything on their own
Had to learn how to work together in a cross-functional team
Perserverance!

It took them about 1-2 years, but they got there...

BUT: You don't have to go through the same pain (nor use 1 year!)

2 mins: installed
2 hours: run your first meaningful tests
2 days: get your team up and running
2 weeks: get a regression suite running in CI
2 months: high coverage for a larger app
2 years: sky's the limit!
It’s possible...

**eBay Denmark - case study**

- **Now:**
  - Small cross-functional team
  - One or two developers implements the story and tests (Unit, Int. and UI tests)
  - Jenkins on a couple of Mac Mini’s (runs in the simulator)
  - A subset of UI tests runs after each commit
  - A subset of UI tests in Xamarin Test Cloud
  - All UI tests runs once every night
HOW?

Which tool should I pick?
How do I get started?
How do I organize test suites for scale and maintainability?
Cross-platform/code reuse?
Automated UI Testing - the robot army

Imagine

• a basement full of programmable robots
• each robot uses a real device that your users also use
• with every change a developer makes to your app, the army tests the important stories
• within minutes the army presents to you a detailed report of the results, including screenshots and video

robot hand
https://creativecommons.org/licenses/by/2.0/
How can you program the robot?

- Lot's of tools out there, e.g.,
  - XCUITest (iOS 9)
  - UIAutomation (iOS)
  - UiAutomator (Android 4.2+)
  - instrumentation (Android)
  - Appium (iOS, Android)
  - Calabash (iOS, Android)
  - KIF 2.0 (iOS), Robotium (Android), Frank (iOS), Espresso (Android), Subliminal (iOS),...
  - Vendor tools (Xamarin.UITest, Monkey talk, eggPlant, ...)

- Here are our criteria for this talk:
  - Open source
  - cross-platform (iOS + Android)
  - general purpose programming language
  - popular & mature & maintained
  - well documented
  - easy to get started
Two Primary Suggestions

Calabash
Automated acceptance testing for mobile apps

https://github.com/calabash/calabash
http://calaba.sh

https://github.com/appium/appium
http://appium.io
DEMO

Getting Started

https://github.com/krukow/agile-2015-samples
Appium

• Pros:
  • Cross platform, open source, active, http-based architecture
  • Supports automating mobile browsers
  • Uses purely platform tools (Android - device control via UiAutomator)
  • Language and Framework agnostic
  • Selenium API will let you leverage some of your Selenium knowledge

• Cons:
  • UiAutomator is Android 4.2+ (Selendroid option but not transparent)
  • Uses purely platform tools: Limited by those platform tools
  • Documentation isn't great
  • Language and Framework agnostic (=> harder to get started, find help)
  • Selenium API not for everyone (e.g. XPath, click, back, navigate to, gestures,...)
Calabash

• Pros:
  • Cross platform, open source, active, http-based architecture
  • Opinionated: Ruby + BDD/Cucumber (getting started is trivial, find help)
    • Specialized API & query system
  • Not limited by platform tools (extended power, e.g. calling methods in the app & going around the UI with 'backdoors')
  • All iOS and Android versions

• Cons:
  • Opinionated (not for everyone) - Ruby (+Cucumber)
  • Specialized API & query system
  • Documentation isn't perfect
  • Needs to be in-process
Which should I choose?

Choose Appium if you:

- prefer un-opinionated with respect to framework and language
- testing a mobile website in addition to native/hybrid apps
- prefer purity over power
- have existing good Selenium engineers and you're happy with it

Choose Calabash if you:

- prefer an opinionated which focuses on BDD with Ruby and Cucumber
- you prefer power over purity
- you're not experienced with or don't particularly like Selenium

If in doubt: Calabash is probably easier to learn and get started with while Appium is more flexible by making fewer choices for you....
Conclusion

NOW is time to begin test automation on mobile...

• The high-profile companies (and maybe your competitors) are doing it.
• The tools are ready, they are open source and free.
• It takes days, not years to learn.
• Techniques and practices are developed.
• You can choose to tap into much value-add via vendor services.
Q&A

Come talk to me!
Leave Feedback on this talk!
Happy to do 1:1s with anyone to help them get started today!
What's Next?
Cross-platform Testing and Recommended Practices
https://github.com/calabash/calabash/tree/develop/samples/wordpress
Do

- Use the Page Object Pattern
- Use waiting with appropriate timeouts
- Use Accessibility for disabled users!
- work as agile testing teams (e.g. dev adds id as they're writing a test with QA)
- Share code using the programming language constructs and normal programming practices
- consider the cost-benefit ratio of automating each test case

Don't

- Give up :)
- Use Sleeps (when other options exist)
- Don't sacrifice accessibility because of automation.
- Silo
- forget to consider localization
- Copy-paste test or automation logic when creating new test
- try to automate every test
What is Xamarin Test Cloud?

1700+ real, non-jailbroken mobile devices, fully automated

A cloud service providing delightful visual test reports

Automated Tests

Built For Continuous Integration
DEMO
References

- **Calabash**
  - Github: https://github.com/calabash/calabash
  - Sample: https://github.com/calabash/calabash/tree/develop/samples/wordpress
  - API: http://calabashapi.xamarin.com/calabash/
  - Landing: http://calaba.sh/
  - https://github.com/krukow/agile-2015-samples

- **Appium**
  - Github: https://github.com/appium/appium
  - Samples: https://github.com/appium/sample-code/tree/master/sample-code/examples
  - Landing: http://appium.io/
  - https://github.com/krukow/agile-2015-samples

- **Xamarin**
  - Test Cloud: http://xamarin.com/test-cloud
  - Xamarin.UITest: http://developer.xamarin.com/guides/testcloud/uitest/
Q&A

Come talk to me!
Leave Feedback on this talk!
Happy to do 1:1s with anyone to help them get started today!
Automated Testing of Mobile Apps

Karl Krukow
Lead, Xamarin Test Cloud
karl@xamarin.com
Demo notes

* Base Installation
** Xcode
** Android SDK
** Java JDK

ANDROID_HOME is set to "/Users/krukow/android/adt/sdk"
JAVA_HOME is set to "/Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home."

** Debug keystore
** device, or sim/emulator
** Need an app to test :)


** Installation:  
https://github.com/calabash/install --  
Just run the curl command.

Right now just use:  
https://github.com/calabash/calabash

** Scaffold a project

mkdir calabash-sample  
cd calabash-sample  
calabash generate  
calabash resign ../Superheroes.apk  
calabash run ../Superheroes.apk
Feature: Playing the game
  Scenario: Tapping a letter
    Given I'm on the main page
    And I start playing
    Then an image of a super hero is shown
    When I choose a letter
    Then that letter becomes selected

*** How would I go about actually testing this?
The Calabash console is a great interactive environment,
calabash console ..../Superheroes.apk
but you can also use uiautomatorviewer.

~/android/adt/sdk/tools/uiautomatorviewer
** Installation:
http://appium.io/ Download the app
Run the doctor and make sure you've set (.bash_profile)

** Sanity test it.
Set device name (with one connected device - just blank)
Make sure you stop appium when reconfiguring.

Run inspector.
Doesn't Live update so Refresh.
Click around to inspect.
Stop.
Appium Demo notes

*** Setup IDE: We'll use Java and jUnit. Easiest to use JetBrains' IntelliJ.
New Project > Maven > Archetype: maven-archetype-quickstart or
mvn archetype:generate -DgroupId=com.xamarin.sample
   -DartifactId=appium-sample
   -DarchetypeArtifactId=maven-archetype-quickstart
   -DinteractiveMode=false

*** Update dependencies
Update to jUnit 4
Add Appium Java Client
<dependency>
   <groupId>io.appium</groupId>
   <artifactId>java-client</artifactId>
   <version>3.1.0</version>
   <scope>test</scope>
</dependency>
Appium Demo notes

*** Use the Appium inspector to scaffold a test...
Create a Class and setup+tear down.
- Create an instance var AppiumDriver< WebElement> wd;
- Import deps and replace AppiumDriver with AndroidDriver.
- capabilities.setCapability("deviceName", ");
- replace wd.close with  wd.quit();

** Sanity test it
A few gotcha's
-- Restart the server if you get an error about existing session

In Settings. Override existing sessions

capabilities.setCapability("deviceName", ");
**Appium Demo notes**

*** How would I go about actually testing this?
Use the inspector to find ids

```java
wd.findElement(By.id("com.icontrivia.GuessTheSuperheroes:id/btnplay")).click();
WebElement superHero =
wd.findElement(By.id("com.icontrivia.GuessTheSuperheroes:id/imageView"));
assertNotNull(superHero);
wd.findElement(By.id("com.icontrivia.GuessTheSuperheroes:id/button")).click();

AndroidElement firstSolutionRow =
(AndroidElement)wd.findElement(By.id("com.icontrivia.GuessTheSuperheroes:id/gui_answer1"));
String selectedText =
firstSolutionRow.findElementByClassName("android.widget.Button").getText();
assertNotNull(selectedText);
```