Take your mobile app tests to the next level : Continuous Integration

Eing Ong
Staff Software Engineer in Quality, Intuit, Inc.

Session outline

- Mobile tests CI challenges
- Android & iOS basics
- Automation tools
- Demo
- Q & A

CI challenges for mobile tests

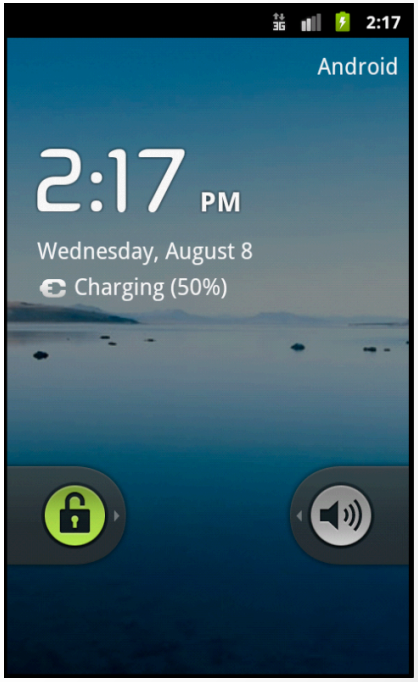
- Mobile platforms
  - OS, resolutions, screen sizes, memory, camera, processor
  - Phone, tablets
- Mobile integration
  - Setup, execution, teardown
  - Test results
- Test automation technologies

Setup - Android

- Command
  $ANDROID_HOME/tools/emulator -avd $avd&
  Useful options: -wipe-data -no-boot-anim -noaudio

- Tip #1: Android emulator plugin
  Start emulator, create snapshot, install, uninstall
  Limitation : Random port for emulator and adb

- Tip #2 : Snapshot images
  emulator -avd <name> -snapshot <name>
  Useful option : -no-snapshot-save
  Limitation : Snapshot image timestamp

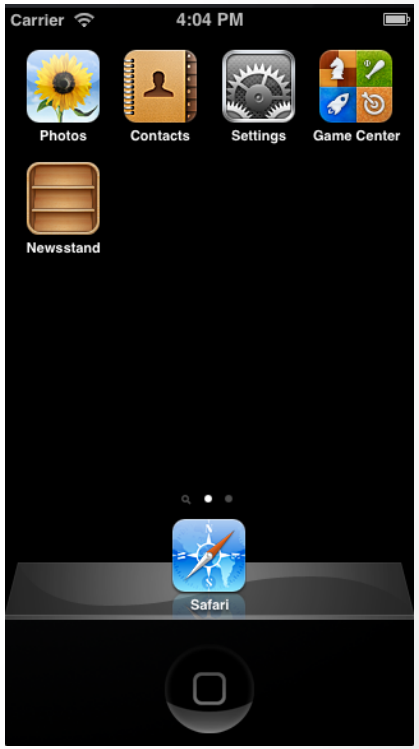


Setup - Android

- Tip #3 : Setting current date
  adb shell date -s `date +"%Y%m%d.%H%M%S"`
- Tip #4 : Restart android debug bridge
  adb devices | grep emulator
  adb kill-server; adb start-server
- Tip #5 : Unlock emulator
  adb shell input keyevent 82
  adb shell input keyevent 4

Setup - iOS

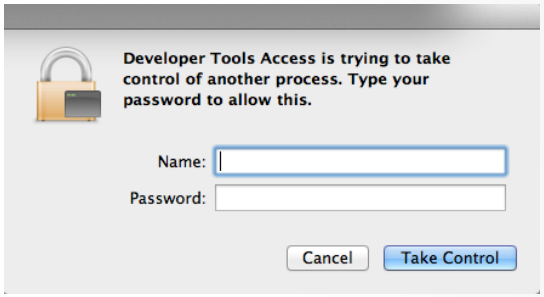
- Tip #1 : Commands
  - open -a "iPhone Simulator.app"
  - iphonesim launch <app name> [sdkversion]
  - instruments
    - –t <automation tracetemplate> <app name>
    - -w <deviceID>
    - -e UIASCRIPT <scriptFilePath>
    - -e UIARESULTSPATH <resultsFolderPath>



Setup - iOS

- Tip #2 – Remove authorization prompt
  - Administrative rights
  - Update /etc/authorization

```
<key>system.privilege.taskport</key>
    <dict>
        <key>allow-root</key>
            <!-- previous value <false/> -->
            <true/>
```



- Tip #3 – Clean cache, preferences, SQLite
  - ~/Library/Application Support/iPhone Simulator/{SDK}/Applications/<uuid>
    Library: Preferences, Caches
    Documents: sqlite3 <app>.sqlitedb

Install

- Android
  - adb install <apk file>

    Multiple installs : find builds -name '*.apk' -exec adb install "{}" \;
  - adb uninstall <package name>
- iOS
  - No separate command needed

Launch

- Android

  adb shell am start

  -n <activity>

  [-*a* android.intent.action.MAIN]

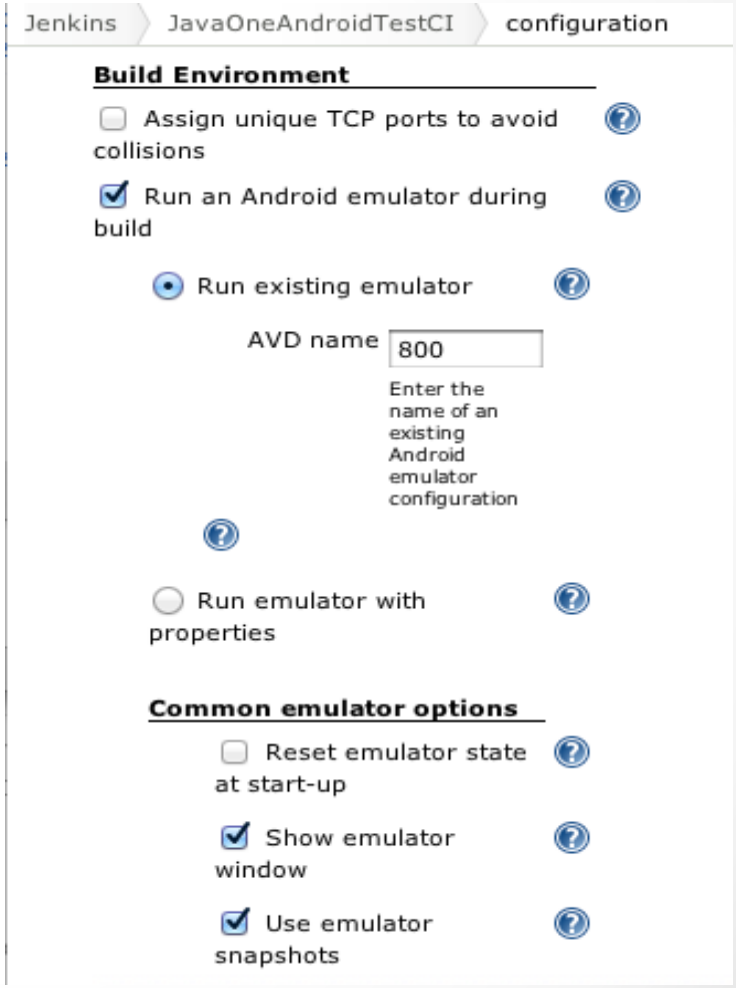
  [-c android.intent.category.LAUNCHER]

  [-c android.intent.category.DEFAULT]

- iOS
  - Similar to setup/startup

Mobile tests in CI

- Build
- Start emulator
- Uninstall & install
- Launch
- Execute tests
- Publish results



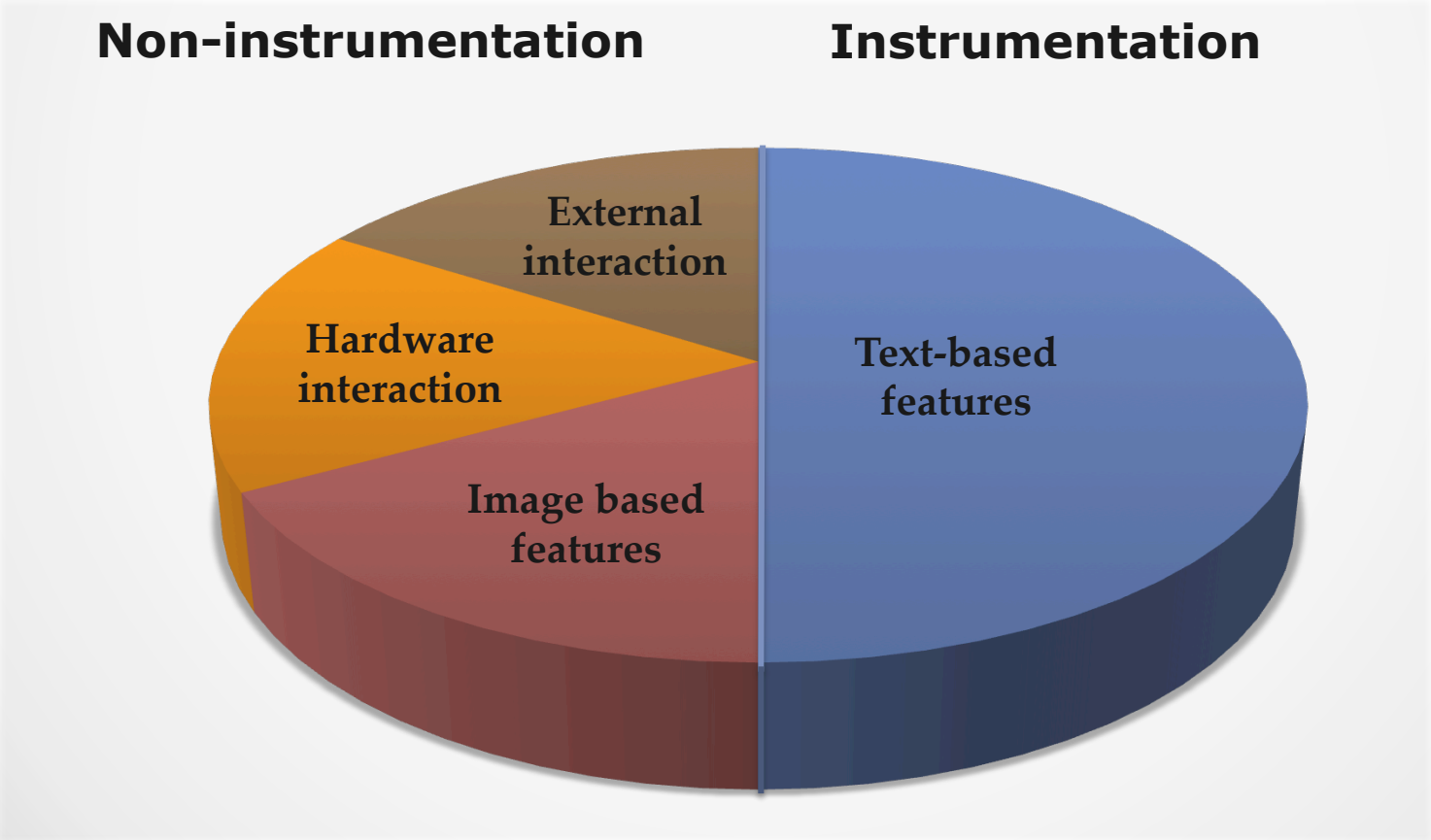
Mobile automation technologies

- Two categories
  - Instrumented technique
  - Non-instrumented technique
- What is instrumentation?
  - Tests are compiled with the app
  - Tests are installed & launched with the app
  - Source code is required and may need to be modified
  - Only one app can be executed at a time
  - White box

Advantages of both techniques

Non-instrumentation	Instrumentation
• Device platform agnostic • Test code reuse • Test language & test harness autonomy • Support for Multi-applications testing Custom UI elements Database/server API assertions Use of external libraries (e.g. image manipulation)	• Elements can be accessed • Debugging ease • Test verification ease • Reduce tools dependencies • Support for Installing application Launching application Cleanup (kill application) Test execution on device Code coverage

Which technique should I use ?



Mobile automation tools

Mobile OS	Non-instrumentation	Instrumentation
Android	eggPlant, Sikuli, MOET, MonkeyRunner	Robotium, Calabash, MonkeyTalk
iOS	eggPlant, Sikuli, MOET	UIAutomation, KIF, iCuke, Frank, UISpec, Zucchini, Bwoken, Calabash, MonkeyTalk

MonkeyRunner

- API for controlling an Android device/emulator outside of Android code - *developer.android.com*
- Actions
  - Multiple emulators & devices interaction
  - Captures screenshots
  - Send keystrokes, gestures
  - Start/stop/reconnect emulator
  - Configure logging

Using MonkeyRunner in Java

- Install Android SDK, tools & platform tools
- Import Android jars & dependencies
  - android.jar, androidprefs.jar, ddmlib.jar, guavalib.jar, hamcrest.jar, junit.jar, jython.jar, monkeyrunner.jar, sdklib.jar
- Important classes
  - com.android.monkeyrunner.adb.AdbBackend
  - com.android.monkeyrunner.adb.AdbMonkeyDevice
  - com.android.monkeyrunner.MonkeyImage
  - com.android.monkeyrunner.MonkeyManager

Using MonkeyRunner in Java

```
public static AdbBackend adbConn = new AdbBackend();
public static AdbMonkeyDevice device;

device = (AdbMonkeyDevice) adbConn.waitForConnection(timeout, deviceId);

device.shell(" am start -n " + activity);
device.shell("kill " + pid);
device.press(KEYCODE_MENU, TouchPressType.DOWN_AND_UP);
device.touch(x, y, TouchPressType.DOWN_AND_UP);
device.type("Hello World");
```

For more details, see com.intuit.moet.Android.java (github.com/eing/moet)

Sikuli

- Visual technology to automate and test GUI using images - *sikuli.org*
- Platform and OS agnostic
  - Controls on desktop
  - Controls mobile simulators and devices (via VNC)
- Actions
  - Captures screenshots
  - Detects screen changes
  - Send keystrokes, gestures
  - Finds image, image OCR

Using Sikuli in Java

- Install Sikuli-IDE.app
  - Import sikuli-script.jar
  - Start using org.sikuli.script.*
- Important classes
  - org.sikuli.script.App
  - org.sikuli.script.Region
  - org.sikuli.script.Screen
  - org.sikuli.script.ScreenImage

Using Sikuli in Java

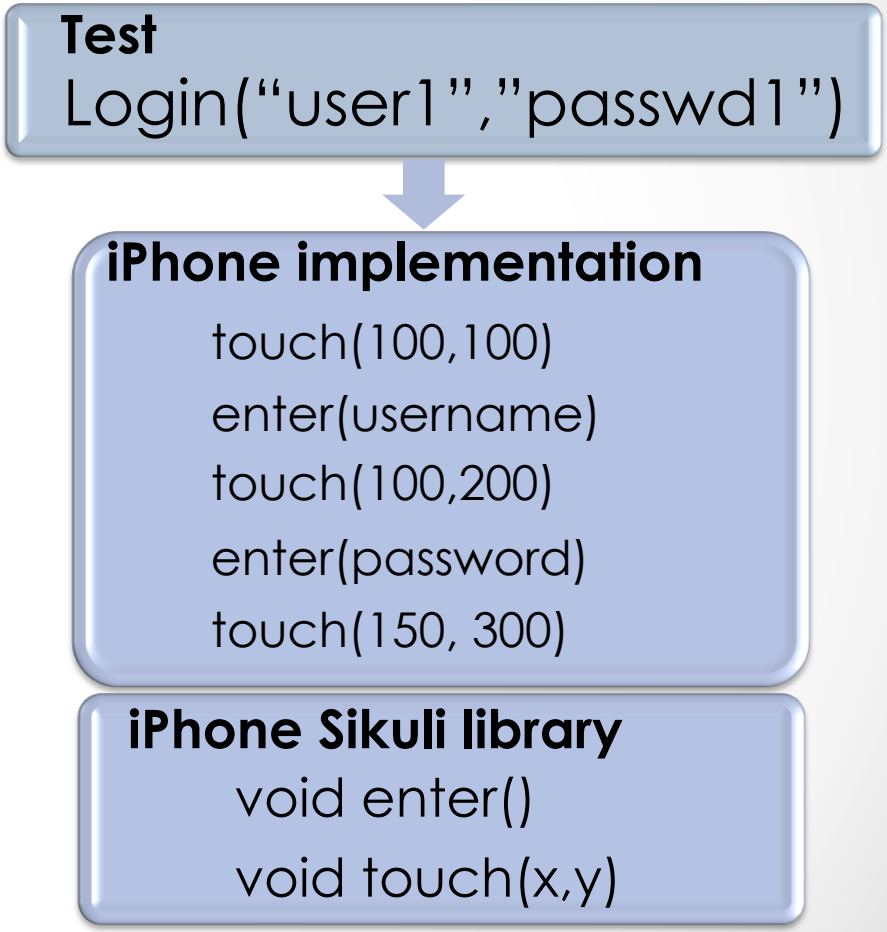
```
public static App app = new App(settings.iphoneExec);
public static Screen screen = new Screen();

screen.setRect(app.window());
region = new Region (screen.getRect());

region.click(new Location(x,y), 0);  // Tap screen
region.click("homebutton.png", 0);  // Tap on image
region.text();                      // Get screen text
region.type(null, inputString, 0);  // Type in focused field
screen.capture(region.getRect());   // Capture screenshot
```

For more details, pls see com.intuit.moet.iPhone.java (github.com/eing/moet)

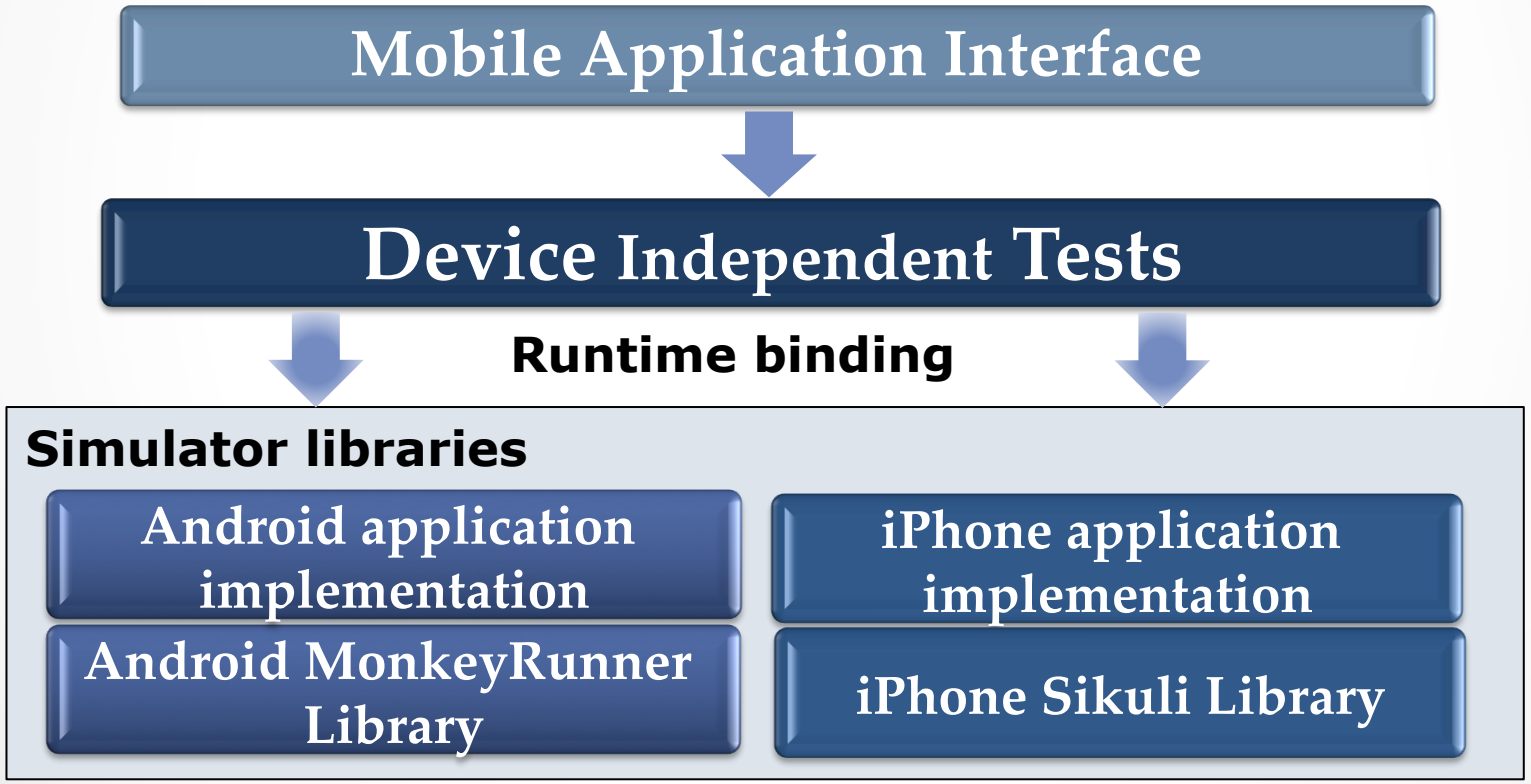
MOET

github.com/eing/moet

- *Think design*
  - Creational pattern
- *Think reuse*
  - Device independent tests
- *Think One*
  - IDE, test language, test harness

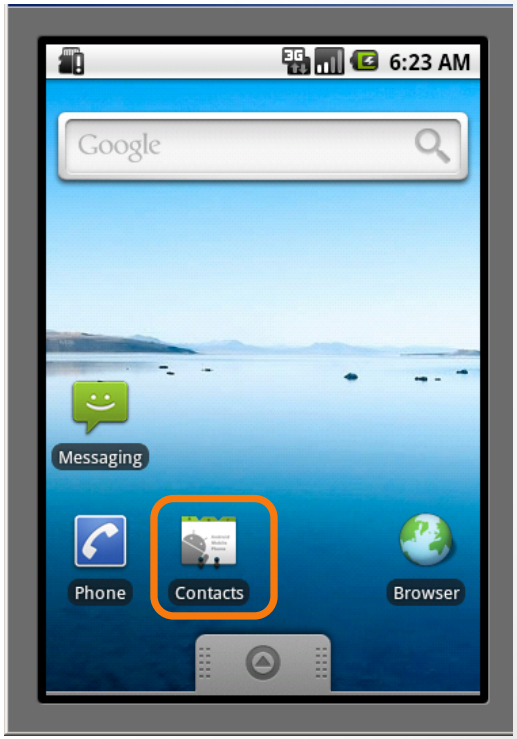


Test architecture using MOET



Demo

- Run a mobile test in CI
  - Android
    - MonkeyRunner, MOET test
    - Install app, launch app, run test, stop app
  - iPhone
    - Sikuli, MOET test
    - Start simulator, launch app, run test, stop app



Resources

- **Android testing**
  - http://developer.android.com/tools/help/index.html
  - http://www.java2s.com/Open-Source/Android/android-core/platform-sdk/com/android/monkeyrunner/MonkeyRunner.java.htm
  - http://code.google.com/p/robotium/
- **iOS testing**
  - http://sikuli.org/docx/faq/030-java-dev.html
  - https://github.com/jhaynie/iphonesim/
  - http://sikuli.org/doc/java-x/index.html?org/sikuli/script/Region.html
- **MOET**
  - https://github.com/eing/moet/tree/master/java/src/com/intuit/moet
- **Jenkins**
  - http://jenkins-ci.org/

Q & A

Thank you

Thank you for your session survey& feedback !

For more details on this presentation,
please contact
@eingong / eing.ong@intuit.com

Jenkins Premier

1. Start Jenkins
   - java –jar jenkins.war (or drop war file in servlet container)
2. Administer Jenkins
   - http://<hostname>:8080 (add /jenkins for war deploy)
   - Install plugins
     - e.g. Perforce, Maven, JUnit, Android emulator
3. Create a new job
   - Setup source code management
   - Build triggers
   - Build steps
   - Publish test report