7th UCAAT User Conference on Advanced Automated Testing

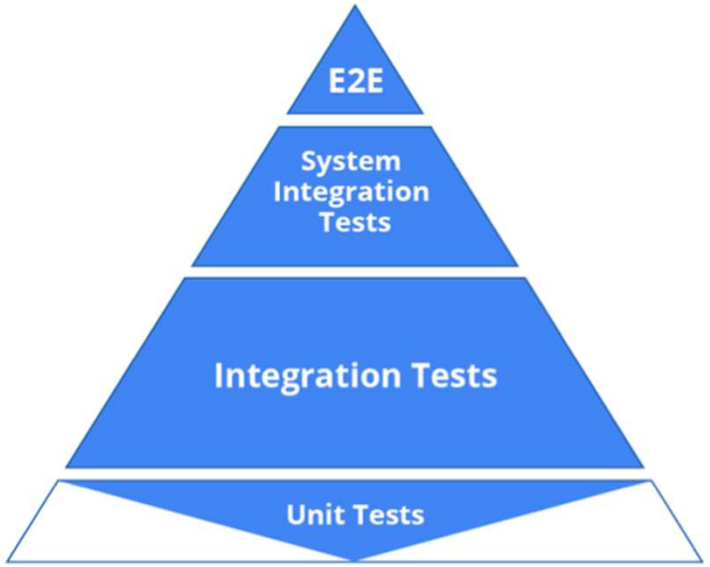
ETSI

A BEGINNER'S GUIDE - UI TESTS FOR IOS

Presented by Václav Vidoň

Agenda

- General background of our apps
- Our tools for automation
- Basic rules for UI testing
- Our CI/CD settings
- Discussion?



General background of our apps

Android

- Kotlin / Java
- ~120k new installs monthly
- ~277k active users monthly
- https://play.google.com/store/apps/details?id=com.skypicker.main

iOS

- Swift / Objective-C
- ~140k new installs monthly
- ~330k active users monthly
- https://itunes.apple.com/cz/app/kiwi-com-cheap-flight-tickets/id657843853

General background of our apps

Android

- Espresso
- UIAutomator
- Kotlin

iOS

- XCTest
- Own utilities (based on XCTest)
- Swift

XCTest

- Used for performing tasks with UI
    - Tapping
    - Typing text
    - Scrolling….
- For iOS only and by Apple
- Works with native apps, multiplatform integrations and webviews
- XCTest
- Contains test recorder (Good, right?)

Espresso & UIAutomator

- Used for performing tasks with UI
  - Tapping
  - Typing text...
- For Android, by Google
- Works with native apps, multiplatform integrations and in-app webviews
- Espresso + UIAutomator
- Contains test recorder and ui inspector (Good, right?)

Multiplatform solutions?s

- Appium/Selenium/Katalon studio…

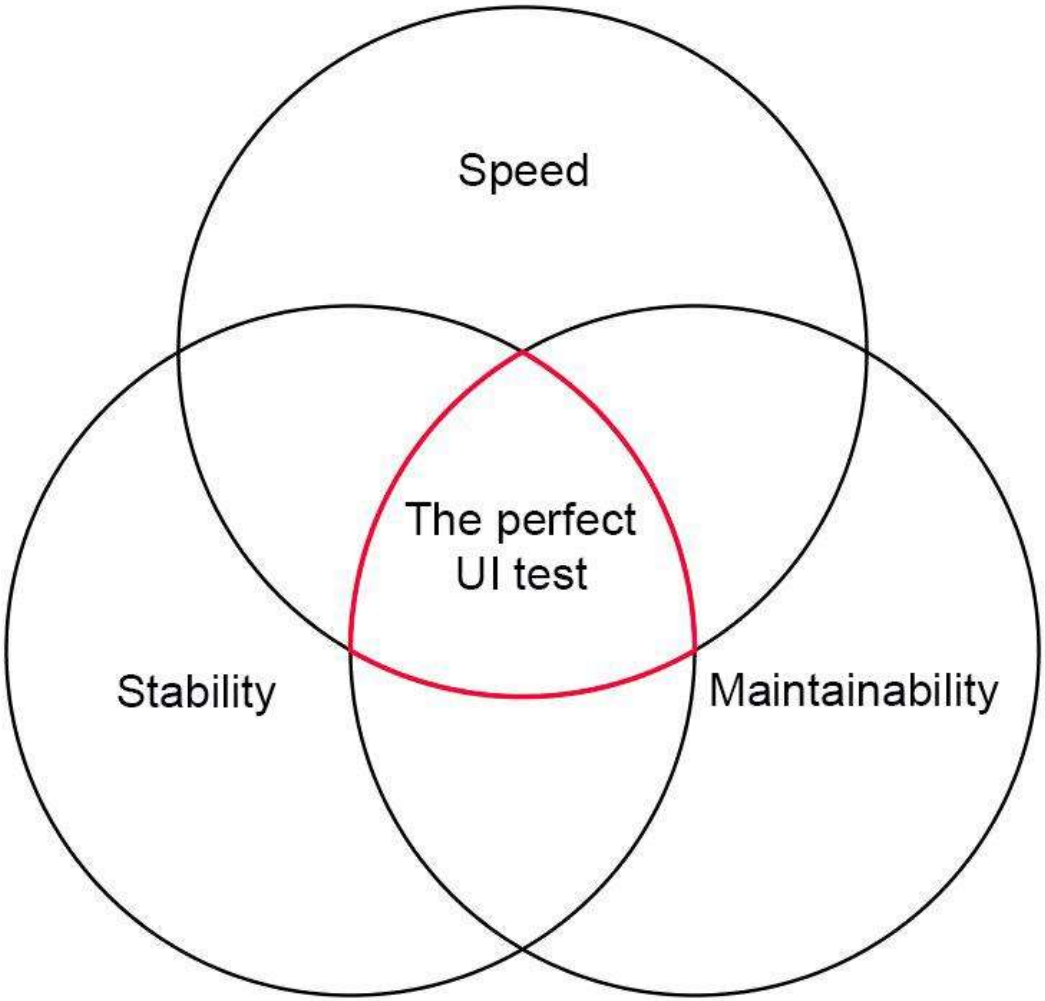
- Still rely on native frameworks - XCTest/Espresso + UIAutomator
- Slower and expensive
- While you can use same test layout (names of methods etc), you still have to have 2 sets of code within these methods
- Not the best CI/CD implementation, compared with xcode line tools/gradlew

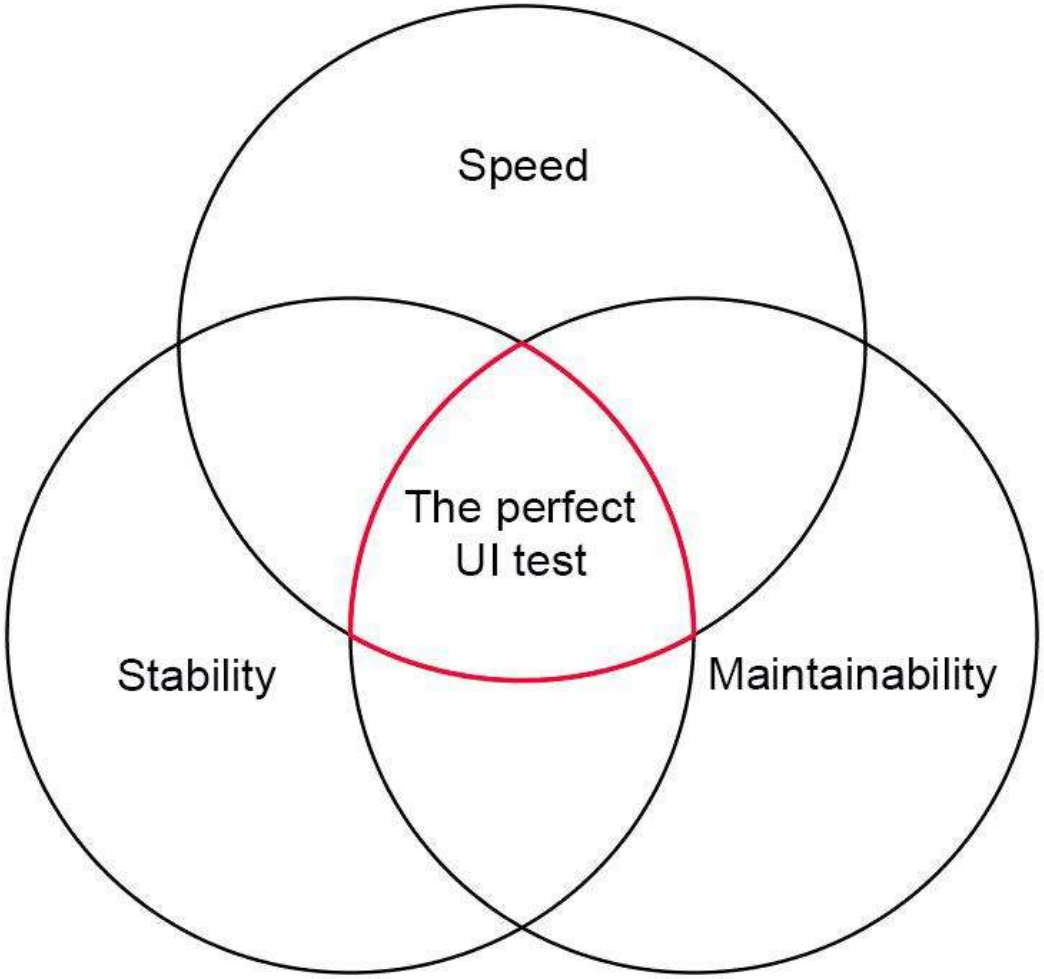
https://medium.com/qaworks/appium-vs-native-frameworks-a-comparison-10c09c6c7e48

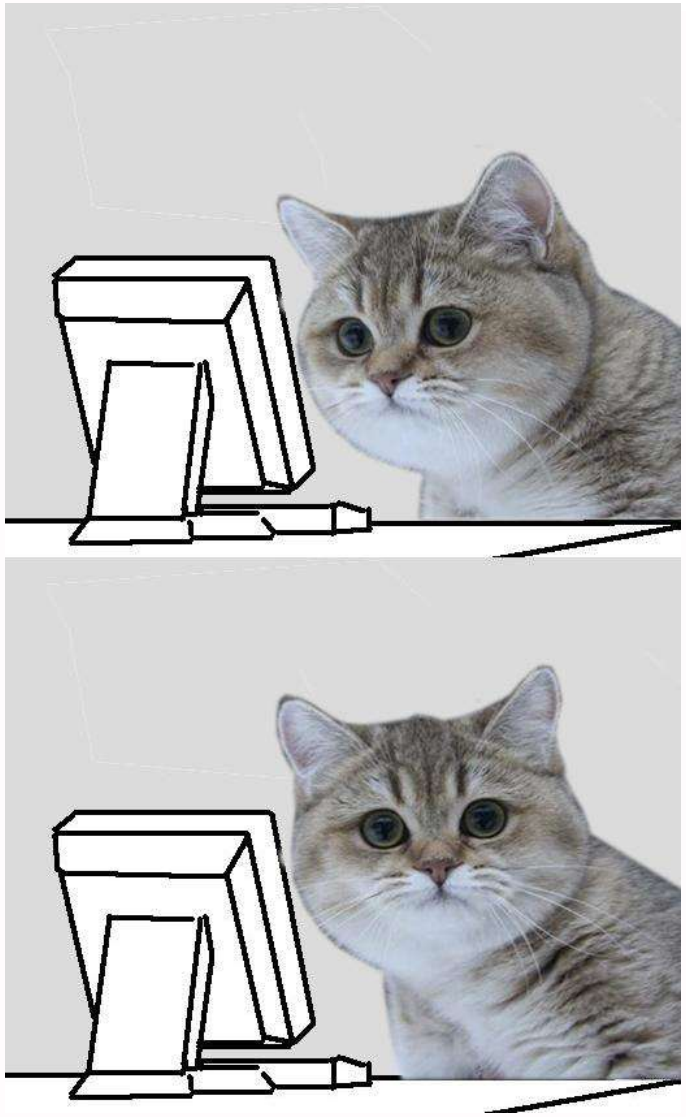
https://stackoverflow.com/questions/46044804/ios-automated-tests-xctest-vs-appium/46064202#46064202

How should a good UI test look like?

- Stability
  - Test
  - Framework
  - Simulators/devices
- Maintainability
  - Easy to debug/fix
  - Framework is evolving with the systems
- Speed
  - Of test/development/fix







Basic rules o a good UI test

- Don't rely on test recorder
- Make a use of console
- Deal with asynchronous events correctly
- Use IDs to identify elements
- Store path to elements in variables
- Use methods for repetitive test code
- Use setUp/tearDown/annotations

```
181     //code generated by test recorder
        func testSomeRandomTest() {
183
184         let app = XCUIApplication()
185         let collectionViewsQuery = app.collectionViews
186         collectionViewsQuery.staticTexts["Help"].tap()
187         app.navigationBars["Help"].buttons["Close"].tap()
188         collectionViewsQuery.staticTexts["Refer a Friend"].tap()
189         app.navigationBars["Kiwi_com.LoginChoiceView"].buttons["Cancel"].tap()
190         app.collectionViews.staticTexts["Messages"].tap()
191         app.navigationBars["Kiwi_com.LoginChoiceView"].buttons["Cancel"].tap()
192     }
```

```
181     //code written manually
182     func testSomeRandomTest() {
183         let app = XCUIApplication()
184
185         let helpStaticText = app.collectionViews.staticTexts["Help"]
186         let messagesStaticText = app.collectionViews.staticTexts["Messages"]
187         let referStaticText = app.collectionViews.staticTexts["Refer a Friend"]
188         let closeButton = app.navigationBars.buttons["Close"]
189         let cancelButton = app.navigationBars.buttons["Cancel"]
190
191         helpStaticText.tap()
192         closeButton.tap()
193         referStaticText.tap()
194         messagesStaticText.tap()
195         cancelButton.tap()
196     }
```

Deal with async events correctly

- Don't use “sleeps”
- Use “waits” instead
- What’s the difference?

Identify elements properly

- Any UI element - Button, textfield...
- How to identify element?
  - Element’s accessibility ID (or other such ID)
  - Element’s string
  - Element’s position (absolute/relative to something)
  - Element's index

```
AccessibilityIdentifier const AccessibilityIdentifierBookingInfantRateFirstOption = @"bookingInfantViewFirstOption";
AccessibilityIdentifier const AccessibilityIdentifierBookingInfantRateSecondOption = @"bookingInfantViewSecondOption";
AccessibilityIdentifier const AccessibilityIdentifierBookingAddAnotherPassengerButton = @"bookingAddAnotherPassenger";
```

```
- (UIButton *)button {
    if (_button == nil) {
        _button = [[RoundedButton alloc] initWithStyle:RoundedButtonStylePrimary];
        [_button setTitle:LocalizedString(@"booking.global.add_another_passenger") forState:UIControlStateNormal];

        _button.accessibilityIdentifier = AccessibilityIdentifierBookingAddAnotherPassengerButton;
    }
```

```
func addInstapassengerWithinBooking(papName: String) {
    let addAnotherButton = app.collectionViews
        .buttons[.bookingAddAnotherPassengerButton]
    let passengerNameString = app.staticTexts[papName]
    let doneButton = app.navigationBars.buttons.element(boundBy: 1)

    app.scrollTo(element: addAnotherButton, direction: .down)
    wait(
        until: addAnotherButton.existsAndHittable,
        timeout: 2,
        or: .fail(message: "Add another pap button is not visible.")
    )
    addAnotherButton.tap()

    wait(
        until: passengerNameString.exists,
        timeout: 5,
        or: .fail(message: "Passenger in instabooking does not exists.")
    )
    passengerNameString.tap()

    wait(until: doneButton.existsAndHittable, timeout: 5, or: .fail(message: "Done button is not visible/hittable"))
    doneButton.tap()
}
```

Store paths to elements in variables

```
func testSomeRandomTest() {
    app.collectionViews.staticTexts["helpID"].tap()
    app.collectionViews.staticTexts["rafID"].tap()
    app.navigationBars.buttons.element(boundBy: 0).tap()

    if app.navigationBars.buttons["Cancel"].exists {
        app.navigationBars.buttons["Cancel"].tap()
    }

    if app.collectionViews.staticTexts["messagesID"].exists {
        app.collectionViews.staticTexts["messagesID"].tap()
    }
}
```

```
func testAnotherRandomTest() {
    let app = XCUIApplication()
    let helpStaticText = app.collectionViews.staticTexts["helpID"]
    let messagesStaticText = app.collectionViews.staticTexts["messagesID"]
    let referStaticText = app.collectionViews.staticTexts["referID"]
    let closeButton = app.navigationBars.buttons.element(boundBy: 0)
    let cancelButton = app.navigationBars.buttons["Cancel"]


    helpStaticText.tap()
    closeButton.tap()
    referStaticText.tap()

    if cancelButton.exists {
        cancelButton.tap()
    }

    if messagesStaticText.exists {
        messagesStaticText.tap()
    }
}
```

Use methods for repetitive test code

```
75      func logInViaEmailFromTabBar(email: String, password: String, expectNotifications: Bool = true) {
76          let signInButton = app.collectionViews.cells.buttons[.signButtonInProfile]
77          let emailLoginButton = app.buttons[.emailLoginButton]
78          let loginWithPasswordField = app.scrollViews.otherElements.secureTextFields[.loginViewPasswordTextField]
79          let loginWithEmailField = app.scrollViews.otherElements.textFields[.loginViewEmailTextField]
80          let submitButton = app.scrollViews.otherElements.buttons[.loginViewSubmitButton]
81
82          switchToTab(.profile)
83
84          wait(
85              until: signInButton.exists,
86              timeout: 2,
87              or: .fail(message: "Couldn't find Sign In button in profile.")
88          )
89          signInButton.tap()
90          emailLoginButton.tap()
91
92          _ = clearTextfieldIfNeeded(textfield: loginWithEmailField)
93          type(text: email, into: loginWithEmailField, pressEnter: true)
94          type(text: password, into: loginWithPasswordField, pressEnter: false)
95          submitButton.tap()
96
97          if expectNotifications {
98              handleNotificationsIfNeeded()
99          }
100     }
```

```
75      func logInViaEmailFromTabBar(email: String, password: String, expectNotifications: Bool = true) {
76          let signInButton = app.collectionViews.cells.buttons[.signButtonInProfile]
77          let emailLoginButton = app.buttons[.emailLoginButton]
78          let loginWithPasswordField = app.scrollViews.otherElements.secureTextFields[.loginViewPasswordTextField]
79          let loginWithEmailField = app.scrollViews.otherElements.textFields[.loginViewEmailTextField]
80          let submitButton = app.scrollViews.otherElements.buttons[.loginViewSubmitButton]
81
82          switchToTab(.profile)
83
84          wait(
85              until: signInButton.exists,
86              timeout: 2,
87              or: .fail(message: "Couldn't find Sign In button in profile.")
88          )
89          signInButton.tap()
90          emailLoginButton.tap()
91
92          _ = clearTextfieldIfNeeded(textfield: loginWithEmailField)
93          type(text: email, into: loginWithEmailField, pressEnter: true)
94          type(text: password, into: loginWithPasswordField, pressEnter: false)
95          submitButton.tap()
96
97          if expectNotifications {
98              handleNotificationsIfNeeded()
99          }
100     }
```

Use setUp/tearDown

```
18      override func setUp() {
19          super.setUp()
20          initialConfiguration(withLanguage:  "en-US", locale: "en-US")
21      }
22
23      override func tearDown() {
24          app.terminate()
25          super.tearDown()
26      }
```

Questions?