SENG 440 Week 11

Testing Android Apps

Ben Adams

Mobile app testing

- Functional testing
- Testing on different operating system versions
- Compatibility with different devices (e.g. sensors)
- Performance on different devices (speed, memory, etc)
- Appearance on different screen sizes
- Accessibility testing

Debugging and profiling

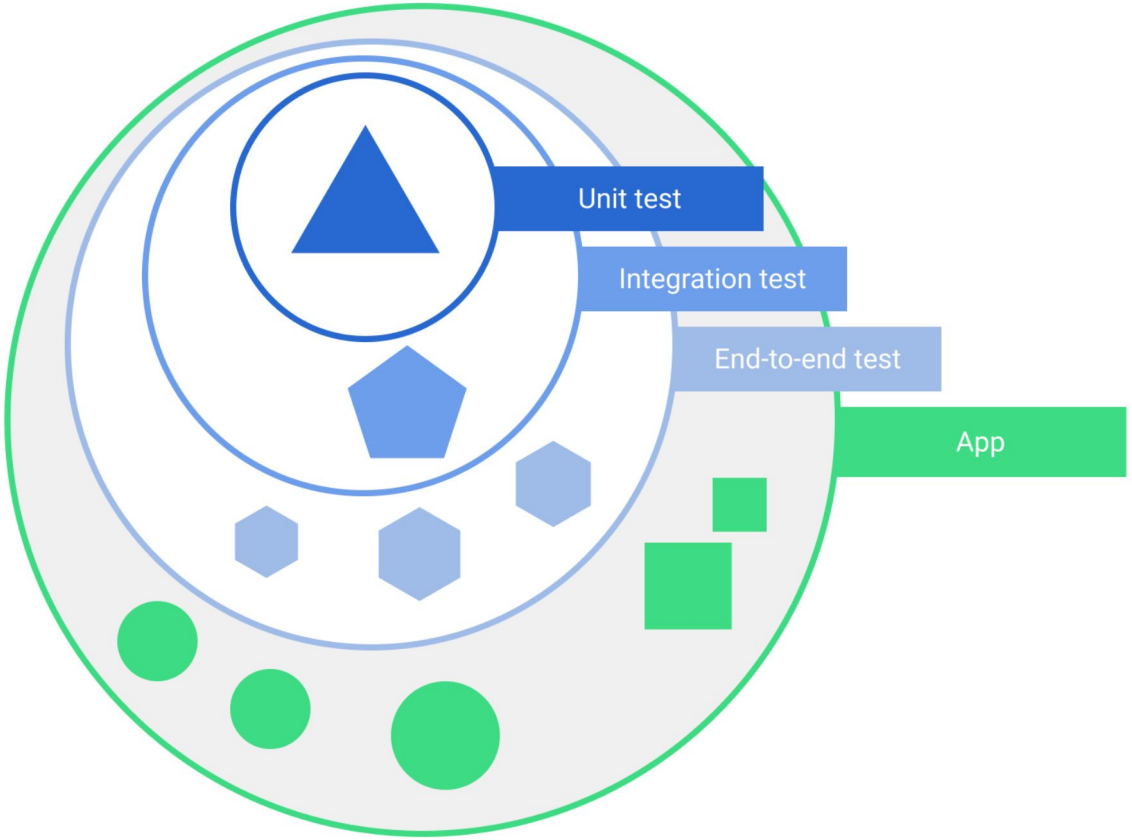
- Log class

  https://developer.android.com/reference/kotlin/android/util/Log

- Stack tracing
- Device file explorer
  - `data/data/`*`app_name`*`/` - app data files on internal storage
  - `sdcard/` - data files on external storage
- Android studio profilers:
  - CPU profiler
  - Energy profiler
  - Memory profiler
  - Network profiler

Testing scope

- Unit tests
- Integration tests
- End-to-end tests



Local tests

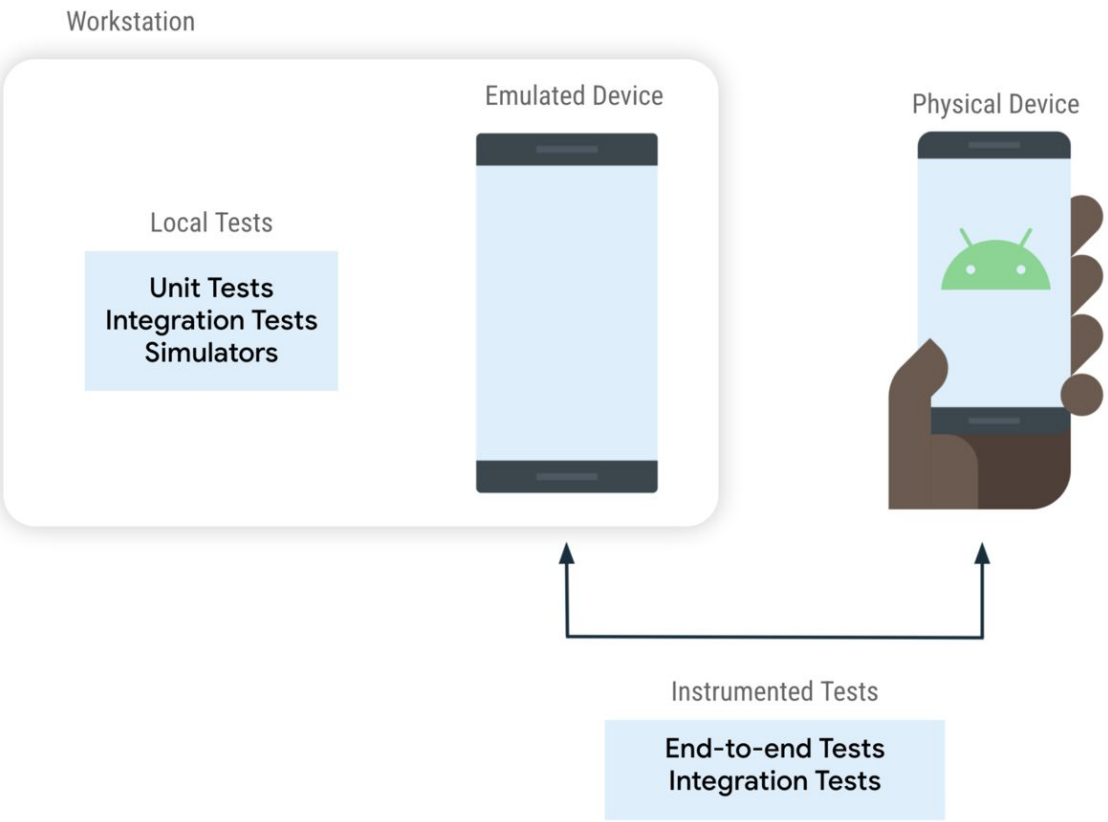
- Limited tests that run on development machine, not device or emulator
- Cannot interact with the Android framework
- Unit tests - execute code and validate result
- Can only access public methods in a class
- Requires dummy data (fakes, test doubles)
- Types of test doubles: [Use test doubles in Android]

```
object FakeUserRepository : UserRepository {
    fun getUsers() = listOf(UserAlice, UserBob)
}

val const UserAlice = User("Alice")
val const UserBob = User("Bob")
```

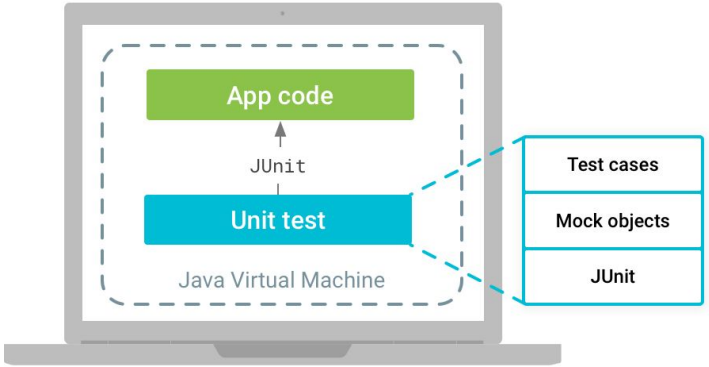
Instrumented tests

- Test application that invokes commands (e.g. UI triggers) on the application that you are testing
- Automated UI tests
- Flaky tests - non-deterministic behavior (turning off system animations can help)

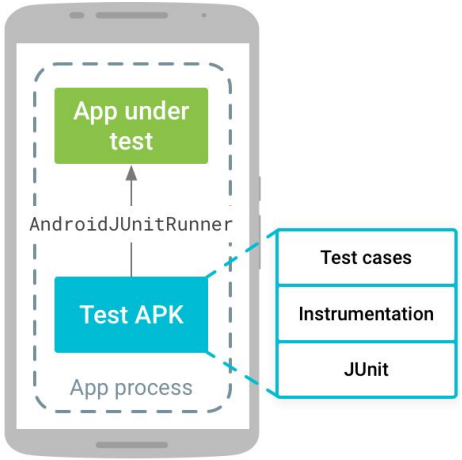


Test environment

- App under test
  - src/main/java
- Local tests
  - src/test/java
- Instrumented tests
  - src/androidTest/java



Local unit test
src/test/java/



Instrumented test
src/androidTest/java/

Androidx test libraries

```
dependencies {
    // Core library
    androidTestImplementation "androidx.test:core:$androidXTestVersion0"

    // AndroidJUnitRunner and JUnit Rules
    androidTestImplementation "androidx.test:runner:$testRunnerVersion"
    androidTestImplementation "androidx.test:rules:$testRulesVersion"

    // Assertions
    androidTestImplementation "androidx.test.ext:junit:$testJunitVersion"
    androidTestImplementation "androidx.test.ext:truth:$truthVersion"

    // Espresso dependencies
    androidTestImplementation "androidx.test.espresso:espresso-core:$espressoVersion"
    androidTestImplementation "androidx.test.espresso:espresso-contrib:$espressoVersion"
    androidTestImplementation "androidx.test.espresso:espresso-intents:$espressoVersion"
    androidTestImplementation "androidx.test.espresso:espresso-accessibility:$espressoVersion"
    androidTestImplementation "androidx.test.espresso:espresso-web:$espressoVersion"
    androidTestImplementation "androidx.test.espresso.idling:idling-concurrent:$espressoVersion"

    // The following Espresso dependency can be either "implementation",
    // or "androidTestImplementation", depending on whether you want the
    // dependency to appear on your APK'"s compile classpath or the test APK
    // classpath.
    androidTestImplementation "androidx.test.espresso:espresso-idling-resource:$espressoVersion"
}
```

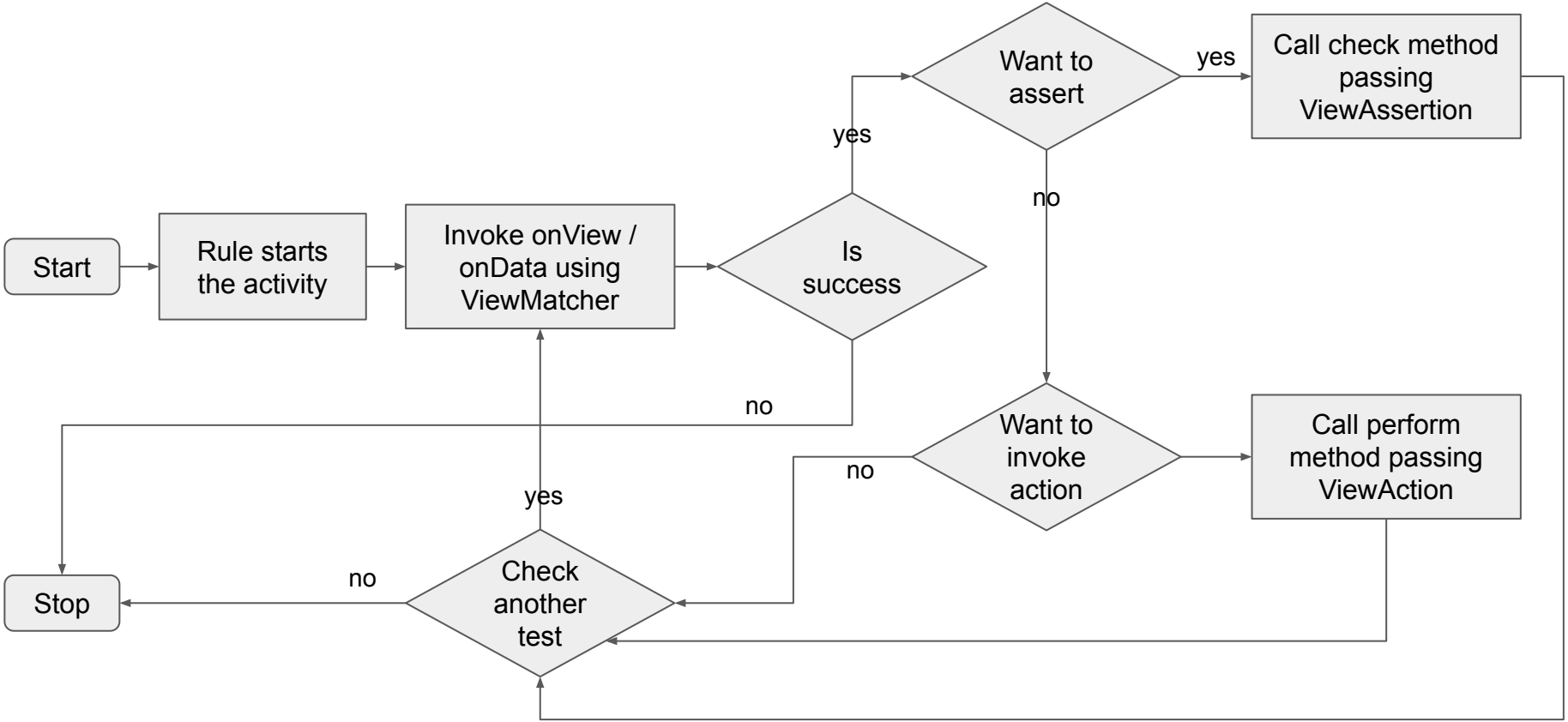
Espresso framework

- Classes
  - JUnit runner
  - JUnit rules
  - `ViewMatchers`
  - `ViewActions`
  - `ViewAssertions`

Synchronization conditions after each execution of interaction with views to prevent flakiness

- The message queue is empty
- No asynchronous tasks executing
- All developer-defined idling resources are idle

GitHub - android/testing-samples: A collection of samples demonstrating different frameworks and techniques for automated testing


Start
Rule starts the activity
Invoke onView / onData using ViewMatcher
Is success
yes
no
Want to assert
yes
no
Call check method passing ViewAssertion
Want to invoke action
no
Call perform method passing ViewAction
Check another test
yes
no
Stop

```
onView(ViewMatcher)
        .perform(ViewAction)
        .check(ViewAssertion);
```

View Matchers

USER PROPERTIES
withId(...)
withText(...)
withTagKey(...)
withTagValue(...)
hasContentDescription(...)
withContentDescription(...)
withHint(...)
withSpinnerText(...)
hasLinks()
hasEllipsizedText()
hasMultilineTest()

HIERARCHY
withParent(Matcher)
withChild(Matcher)
hasDescendant(Matcher)
isDescendantOfA(Matcher)
hasSibling(Matcher)
isRoot()

INPUT
supportsInputMethods(...)
hasIMEAction(...)

UI PROPERTIES
isDisplayed()
isCompletelyDisplayed()
isEnabled()
hasFocus()
isClickable()
isChecked()
isNotChecked()
withEffectiveVisibility(...)
isSelected()

CLASS
isAssignableFrom(...)
withClassName(...)

ROOT MATCHERS
isFocusable()
isTouchable()
isDialog()
withDecorView()
isPlatformPopup()

OBJECT MATCHER
allOf(Matchers)
anyOf(Matchers)
is(...)
not(...)
endsWith(String)
startsWith(String)
instanceOf(Class)

SEE ALSO
Preference matchers
Cursor matchers
Layout matchers

View Actions

CLICK/PRESS
click()
doubleClick()
longClick()
pressBack()
pressIMEActionButton()
pressKey([int/EspressoKey])
pressMenuKey()
closeSoftKeyboard()
openLink()

GESTURES
scrollTo()
swipeLeft()
swipeRight()
swipeUp()
swipeDown()

TEXT
clearText()
typeText(String)
typeTextIntoFocusedView(String)
replaceText(String)

View Assertions

matches(Matcher)
doesNotExist()
selectedDescendantsMatch(...)

LAYOUT ASSERTIONS
noEllipseizedText(Matcher)
noMultilineButtons()
noOverlaps([Matcher])

POSITION ASSERTIONS
isLeftOf(Matcher)
isRightOf(Matcher)
isLeftAlignedWith(Matcher)
isRightAlignedWith(Matcher)
isAbove(Matcher)
isBelow(Matcher)
isBottomAlignedWith(Matcher)
isTopAlignedWith(Matcher)

```
onData(ObjectMatcher)
        .DataOptions
        .perform(ViewAction)
        .check(ViewAssertion);
```

Data Options

```
inAdapterView(Matcher)
atPosition(Integer)
onChildView(Matcher)
```

```
intended(IntentMatcher);
```

```
intending(IntentMatcher)
    .respondWith(ActivityResult);
```

Intent Matchers

INTENT

```
hasAction(...)
hasCategories(...)
hasData(...)
hasComponent(...)
hasExtra(...)
hasExtras(Matcher)
hasExtraWithKey(...)
hasType(...)
hasPackage()
toPackage(String)
hasFlag(int)
hasFlags(...)
isInternal()
```

URI

```
hasHost(...)
hasParamWithName(...)
hasPath(...)
hasParamWithValue(...)
hasScheme(...)
hasSchemeSpecificPart(...)
```

BUNDLE

```
hasEntry(...)
hasKey(...)
hasValue(...)
```

COMPONENT NAME

```
hasClassName(...)
hasPackageName(...)
hasShortClassName(...)
hasMyPackageName()
```

Jetpack Compose testing API

- Same principles as Espresso but some different elements
- Finders
  - Select one or more nodes in the UI hierarchy
  - Uses matchers (similar to pattern matching)
- Assertions
- Actions

Testing your Compose layout - Jetpack

Finders

```
onNode(matcher)
onNodeWithContentDescription
onNodeWithTag
onNodeWithText
onRoot
```

```
onAllNodes(matcher)
onAllNodesWithContentDescription
onAllNodesWithTag
onAllNodesWithText
```

OPTIONS: `useUnmergedTree: Boolean`

Matchers

```
has[No]ClickAction
hasContentDescription[Exactly]
hasImeAction
hasProgressBarRangeInfo
has[No]ScrollAction
hasScrollTo[Index|Key|Node]Action
hasSetTextAction
hasStateDescription
hasTestTag
hasText[Exactly]
is[Not]Dialog
is[Not]Enabled
is[Not]Focused
is[Not]Selected
isHeading
isOff
isOn
isPopup
isSelectable
isToggleable
isFocusable
isRoot
```

HIERARCHICAL

```
hasParent
hasAnyChild
hasAnySibling
hasAnyDescendant
hasAnyAncestor
```

SELECTORS

```
filter(matcher)
filterToOne(matcher)
onAncestors
onChild
onChildAt
onChildren
onFirst
onLast
onParent
onSibling
onSiblings
```

Assertions

```
assert(matcher)
assertExists
assertDoesNotExist
assertContentDescriptionContains
assertContentDescriptionEquals
assertIs[Not]Displayed
assertIs[Not]Enabled
assertIs[Not]Selected
assertIs[Not]Focused
assertIsOn
assertIsOff
assertIsToggleable
assertIsSelectable
assertTextEquals
assertTextContains
assertValueEquals
assertRangeInfoEquals
assertHas[No]ClickAction
```

COLLECTIONS

```
assertAll
assertAny
assertCountEquals(Int)
```

BOUNDS

```
assert[Width|Height]IsEqualTo
assertIsEqualTo
assert[Width|Height]IsAtLeast
assertTouch[Width|Height]IsEqualTo
assertTopPositionInRootIsEqualTo
assertLeftPositionInRootIsEqualTo
getAlignmentLinePosition(BaseLine)
getUnclippedBoundsInRoot
```

Actions

```
performClick
performTouchInput
performMultiModalInput
performScrollTo
performSemanticsAction
performKeyPress
performImeAction
performTextClearance
performTextInput
performTextReplacement
```

TOUCH INPUT

```
click
doubleClick
longClick
pinch
swipe
swipe[Down|Left|Right|Up]
swipeWithVelocity
```

TOUCH INPUT PARTIAL

```
down
moveTo
movePointerTo
moveBy
movePointerBy
move
up
cancel
```

ComposeTestRule

```
@get:Rule
val testRule =
    createComposeRule()
```

```
setContent { }
density
runOnIdle { }
runOnUiThread { }
waitForIdle()
waitUntil { }
awaitIdle()
[un]registerIdlingResource()
mainClock.autoAdvance
mainClock.currentTime
mainClock.advanceTimeBy()
mainClock.advanceTimeByFrame()
mainClock.advanceTimeUntil { }
```

AndroidComposeTestRule

```
@get:Rule
val testRule =
    createAndroidComposeRule<Activity>()
```

```
ComposeTestRule.* +
activity
activityRule
```

Debug

```
onNode(...).*
```

```
printToString()
printToLog()
captureToImage()
```

UI Automator

- Used for cross app UI testing
- Provides `UiDevice` class to perform operations on the device not tied to a single app, e.g. click home button
- Can access installed apps, e.g. system functions
- Look up UI components with descriptors (works best with Android accessibility features)

[Write automated tests with UI Automator | Android Developers]