Mobile Testing

12 June 2020

Christopher-Robin Jonsson
Fredrik Holmberg

Introduction to Mobile Testing

Testing provides:

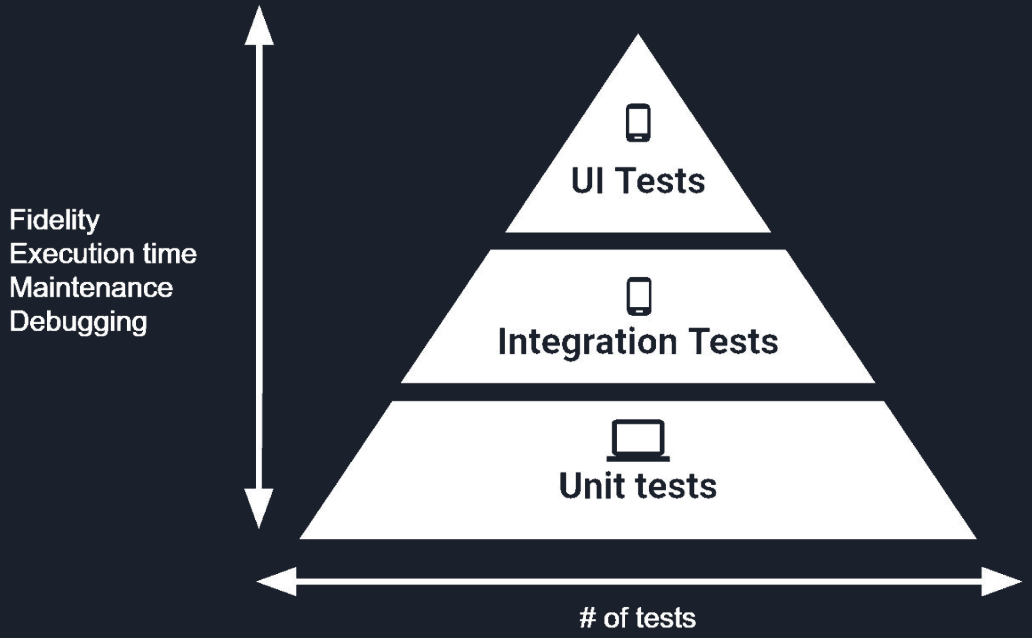
- Rapid feedback
- Failure detection at an early stage
- Safe code refactoring.
- Stable development velocity

Test the application on:

- Real device
- Virtual Device
- Simulated Device

Levels of the Testing Pyramid

- Small tests - Unit tests
- Medium tests - Integration tests
- Large tests - UI tests
- General recommendation from Google:
  - 70% small, 20% medium and 10% large.
- Each test level increase in fidelity and execution time



User Interface Tests (UI Tests)

- Large Test
- Test the users interaction with the application
- UI tests runs on a real or simulated device
- UI test is in the directory “androidTest”
- Android Espresso Testing Framework

Integration Tests

- Medium Test
- How the application interacts with the hardware
- Can access a local database, file system or network

Unit Tests

- Run to detect setbacks after the code has been changed
- Isolate parts of code
- Mock objects that mimics the behaviours of real objects
- Testing Frameworks
  - Robolectric - Framework that brings fast and reliable unit tests to Android
  - JUnit - Framework for to write repeatable unit tests.
  - Mockito - Mocking framework for unit testing

Local Unit Tests

- Part of the “Small Test”
- Run locally on the user machine
- Run on Java Virtual Machine
- Short runtime

Example of test method for local unit test

```
import com.google.common.truth.Truth.assertThat
import org.junit.Test

class EmailValidatorTest {
    @Test
    fun emailValidator_CorrectEmailSimple_ReturnsTrue() {
        assertThat(EmailValidator.isValidEmail("name@email.com")).isTrue()
    }
}
```

Instrumented Tests

- Runs on an emulator or physical device
- Can use AndroidX
- Slower but more faithful than local test

Test Suite Example

```
import com.example.android.testing.mysample.CalculatorAddParameterizedTest
import com.example.android.testing.mysample.CalculatorInstrumentationTest
import org.junit.runner.RunWith
import org.junit.runners.Suite

// Runs all unit tests.
@RunWith(Suite::class)
@Suite.SuiteClasses(CalculatorInstrumentationTest::class,
        CalculatorAddParameterizedTest::class)
class UnitTestSuite
```

Conclusion

Many different ways to test

Crucial part of software development

Without testing it is hard to know what needs to be fixed.