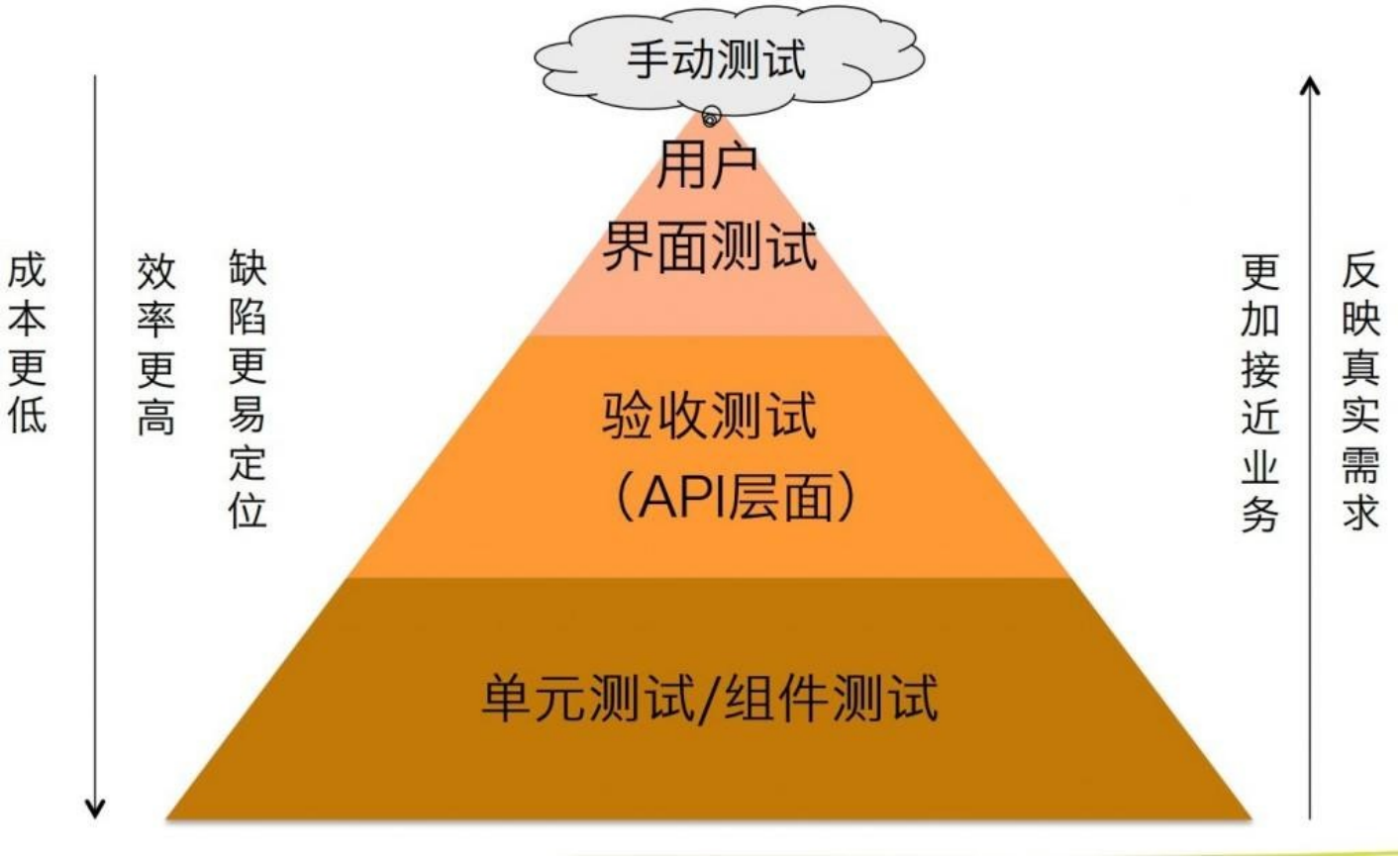
深入浅出Android 自动化测试

测试金字塔



一、常用工具

- monkey、EasyMonkey、MonkeyRunner、Mockito、powermock、AssertJ、UIAutomator(2)、Roboletric、Robotium、Espresso、Selendroid、Appium、Calabash、Athrun
- Google
- [Google 测试支持库： AndroidJUnitRunner、Espresso、UI Automator
- Google官网培训课程Testing Apps on Android

二、分类

1、黑/白 分类

- 有些工具是介于黑/白盒之间的 `灰盒`，不单独划分，归纳到更偏重的一方中。

白盒	黑盒
JUnit AndroidJUnitRunner Mockito powermock Roboletric	money MonkeyRunner UIAutomator Robotium Espresso Selendroid Appium Calabash Athrun 人肉、硬件辅助测试

2、根据具体实现分类

2.1、基于Android Java Instrumentation框架

`Instrumentation框架` 通过将主程序和测试程序运行在 `同一个进程` 来实现这些功能。Google针对Android的环境，扩展了业内标准的JUnit测试框架。

- Robotium
- Espresso（google）
- Selendroid(资料较少，社区活跃度也不大)
- Athrun（淘宝）

2.2 、基于Android lib层的各种命令(sendevent,getevent, monkey, service)，然后用各种语言封装

- MonkeyRunner

- EasyMonkey
- UiAutomator(抓去App页面上的控件属性,不支持Hybird App、WebApp)
- Appium(目前最火的框架)

3、单元测试

- JUnit
- AssertJ: 断言神器，用于替换JUnit的Assert API 拓展库
- Assert语法的拓展库:开源的Assert语法拓展库，它提供了方便、强大的情景匹配API，不仅支持Java，也支持其他多种语言。
- AssertJ-Android:Square针对AssertJ的一个拓展，提供了对Android各类控件的情景检查。
- Robolectric：通过实现一套JVM能运行的Android代码，从而做到脱离Android环境运行测试
- RoboSpock 合并了 Robolectric 和 Spock （适合行为驱动开发）

4、性能测试小工具

- Emmagee 监控指定被测应用在使用过程中CPU、内存、流量资源消耗的性能测试小工具

5、`人肉`、硬件辅助测试

- `人肉`： 测试人员通过手动点击、肉眼观察测试结果。
- 硬件辅助测试：硬件辅助视频录像、机械手臂点击（xx抢红包）等。

三 、工具差异

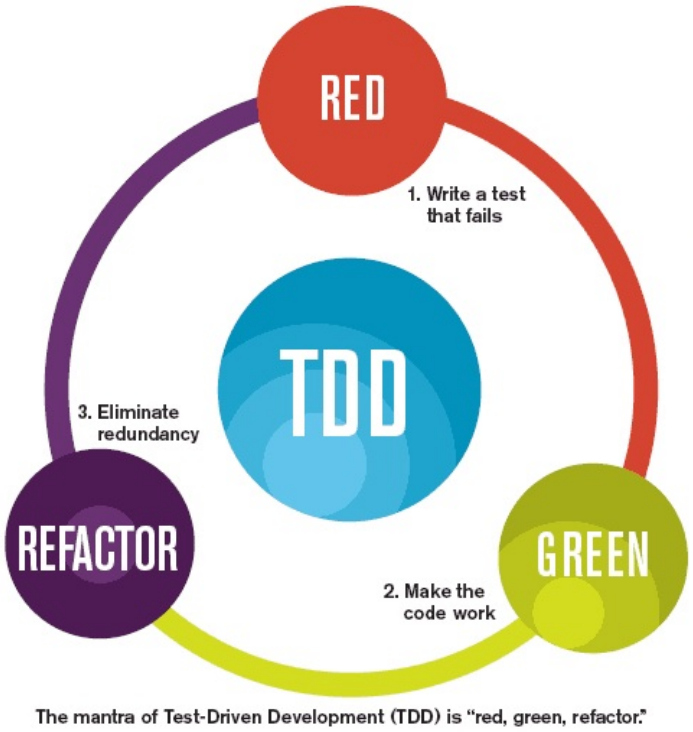
* 主流黑盒工具差异

	UiAutomator	Appium	Robotium	Espresso
支持语言	Java	Java、python、Ruby、PHP等	Java	Java
是否需要签名	不需要	不需要	不需要	不需要
是否支持跨应用测试	支持	不支持	不支持	不支持
是否跨平台	仅支持Android平台	Android、IOS都支持	仅支持Android平台	仅支持Android平台
是否支持WebView	不支持	支持Native App、Hybird App、Web App	不支持	不支持

- Robotium、Espresso、robolectric、Selendroid： 基于Android instrumentation框架，与被测App在同一进程。执行速度快。
- 据统计Espresso的速度是Robotium的5.7倍，很大程度上替代了Robotium。
- `Robolectric`: 只在Java虚拟机中运行，速度很快，在API上无法和Espresso相比，但速度有很大优势，适合单元测试，更适合`TDD`。
- MonkeyRunner：不需要源码、对google官方工具money的封装。
- UIAutomator:不需要source code、google提供的黑盒测试工具与APP在不同进程,但不支持WebView。
- `Appium`、Calabash： `Appium`目前黑盒测试中应用最多的框架。不需要source code，跨平台（ios/andorid），支持WebWiew 和多种语言编写测试脚本，但因为与被App在不同进程，执行速度慢。
- Athrun：淘宝提供的自动化测试工具,跨平台（ios/andorid），包括自动化测试框架，持续集成体系。

四、实践

- World-Class Testing Development Pipeline for Android (中文翻译-世界级的Android测试开发流程)
- Mockito和Roboletric进行Android单元测试
- Espresso和UIAutomator - 完美的结合
- Google+ 团队的 Android UI 测试
- Android测试驱动开发(TDD)



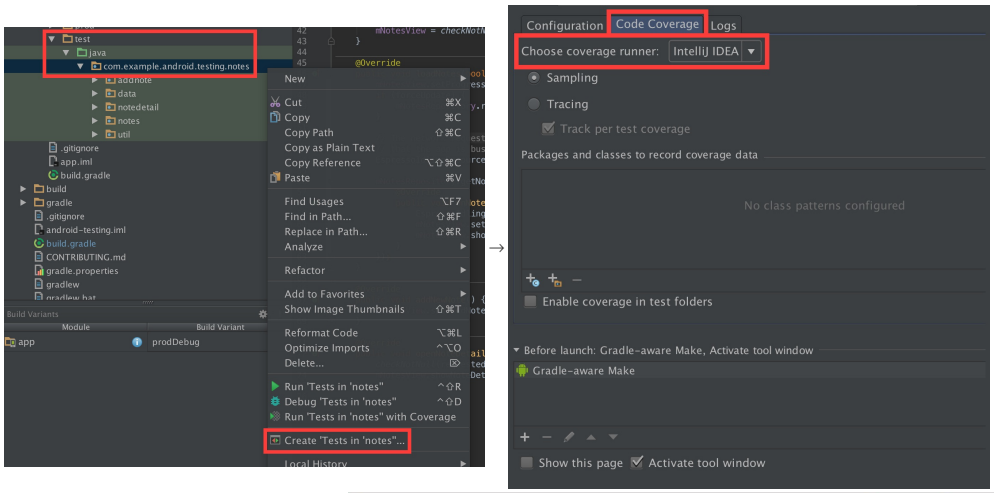
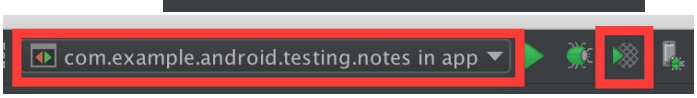
TDD

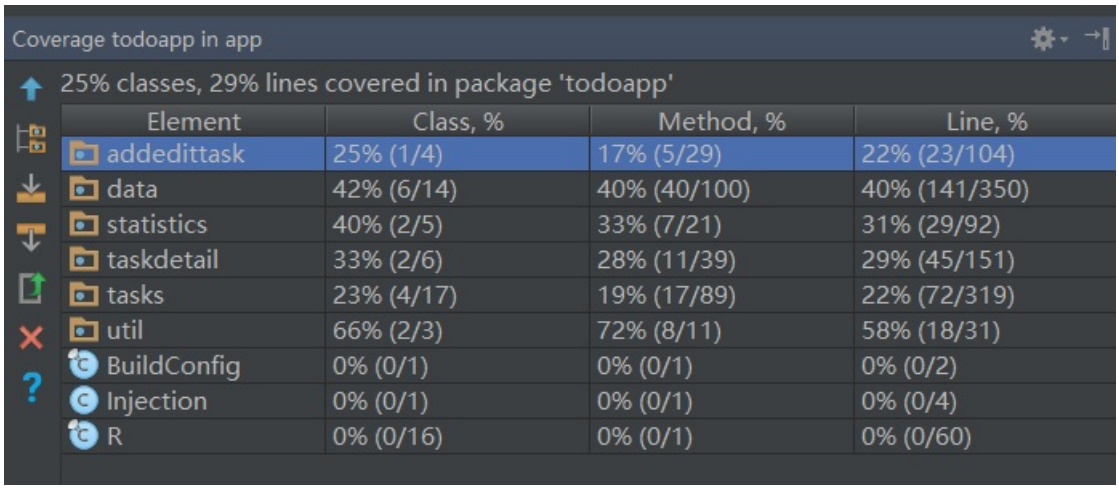
- 通过mock来减少模块依赖（意味着需要定义大量接口，面向接口编程）

五、代码覆盖率

代码覆盖（`Code coverage`）是软件测试中的一种度量，描述程式中源代码被测试的比例和程度，所得比例称为代码覆盖率。

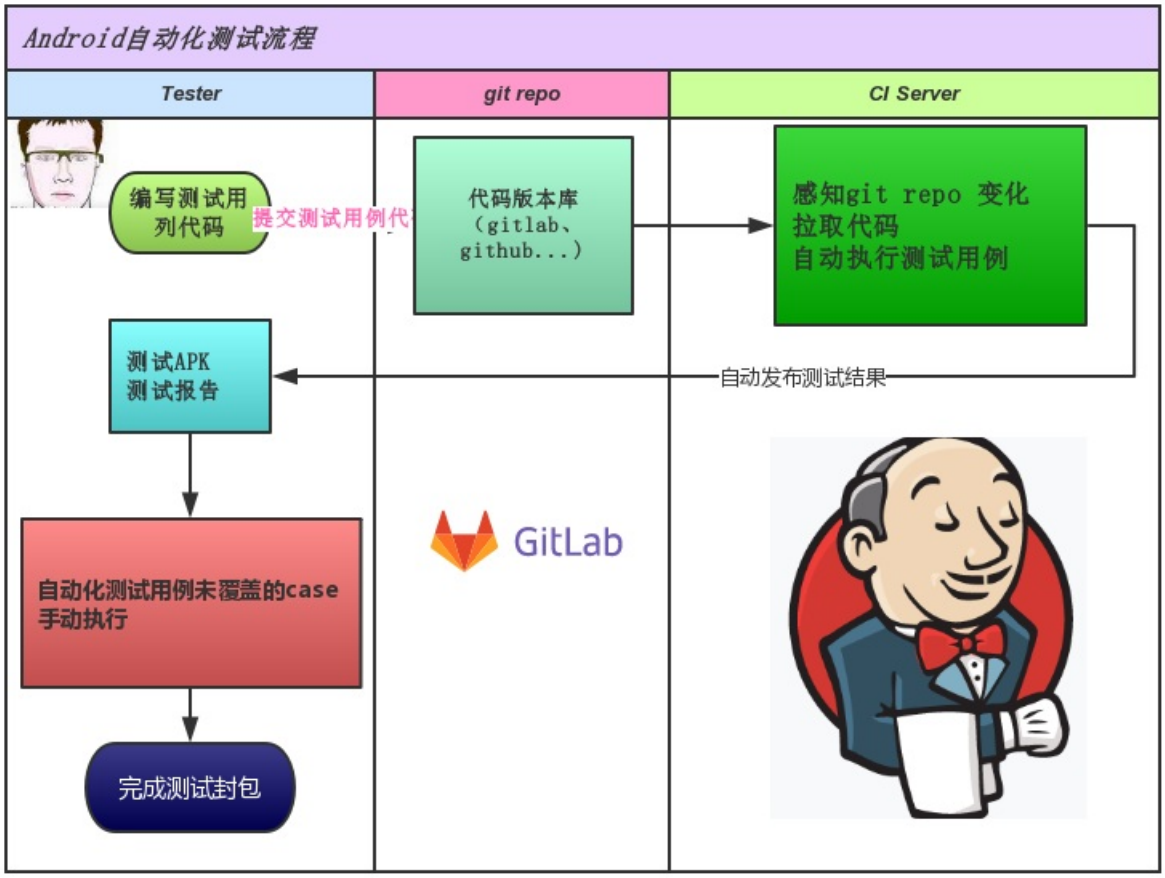
Java	JavaScript	.Net	C/C++	Ruby
JaCoCo Emma Coverlipse Cobrertura	JSCoverage	ColverNET NCover PartCover AQtime	Bullseye Coverage	rcov

- JaCoco 配置Demo
- AndroidStudio自带覆盖率检测工具使用：
  - 
  - 

- 

六、自动化

- Android Unit Testing in Android Studio and CI Environments
- 所有测试应以自动化为主，无法自动化的测试用例，要人工完成。本人认为命令行工具的使用是自动化的基础，推荐阅读为什么优秀的程序员喜欢命令行。
- 持续集成CI，保证代码集成到主干或发布到测试人员手中之前，必须通过自动化测试。只要有一个测试用例失败，就不能集成。
- Jenkins
- Travis
- Codeship
- Strider

- 

七、总结

- 单元测试，所有测试方法中最接近源码实现的一种，但因Android系统的复杂性，不能完全的覆盖所有真实的case,需要结合其它测试方法共同协作提升软件质量。
- `TDD` 的核心思想是测试先来，实现后来。但如何测试并没有规定非要用代码，所以应根据实际情况，选择最佳的测试手段。
- 实际测试时应该结合多个框架使用，单元测试 + 集成测试 + UI测试。
- 各种测试方法都有一个共同的目标，保证最终的软件质量。
- 测试的深度和产品业务需求之间需要平衡。不能因为过渡追求完美的测试而阻碍业务的发展，但也不能因为遗漏重要的代码分支逻辑，给最最终的产品带来重大线上bug。

author: 程涛涛
date: 2017-8-10
email: taotao.cheng@56qq.com