咪咕阅读客户端质量提升实践

咪咕数字传媒有限公司 张琦

2017年11月

基本介绍 1

基本介绍

个人介绍

张琦

曾担任过阿里的测试开发，参与过淘宝的营销平台、手机淘宝、喵街、支付宝国际金融网络等项目的测试。

现担任咪咕阅读客户端的测试负责人。

负责产品介绍

质量提升

2.1 质量提升

◆ 背景

“咪咕阅读”的前身是中国移动旗下的“和阅读”。因各种原因，早期的咪咕阅读产品质量一般，稳定性差；版本迭代周期长，经常延期。

◆ 目的

1.加快迭代速度，每个版本每两周迭代一次以上。
2.提高产品的质量及稳定性。

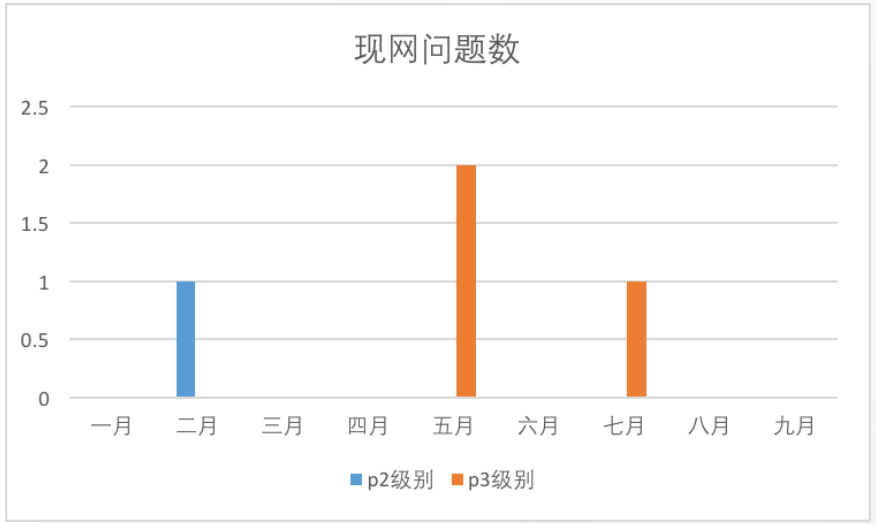
◆ 思路

1.结合先进互联网公司及敏捷测试的经验，改善功能测试流程。
2.性能测试（测试→监控）
3.自动化测试（结果导向）

质量提升 2.2

质量提升

- 稳定性提升：安卓和iOS客户端的Crash率都低于XXX%，比两年前降低了近100倍。比业内领先的竞品（腾讯出品）低20倍。
- 质量提升：全年出现3次P3级别、1次P2级别的生产环境的问题。数量是去年的1/10。
- 迭代周期缩短：每个产品每半月迭代1次，每天都有小型发布。



功能测试 3.1

测试是什么？不忘初心

测试是什么？
自动化测试？性能测试？DevOps？Docker？敏捷？安全测试？全栈工程师？
这些只是手段，不是目的。
目的是保证项目按时保质保量上线。

测试是什么？就是在开发快完成时对程序找bug吗？
------其实不然，就好像捕鱼一样，讲究季节，阳光，水流，甚至渔网洞的大小的使用都直接影响到捕鱼的效果。测试也是一样，不仅仅只是找bug而已，还需要有计划的进行，还需要使用一些技术手段。同时，大家都知道软件缺陷发现越早，修改的成本越低。

功能测试 3.2 功能测试

用例设计

1 用例设计要有技巧，不仅仅是依靠“经验”，经验有时是把双刃剑（有时会束缚你的思维）。
边界值、等价类、因果图、错误推测、流程分析、正交验证。。。
基本所有测试都知道，但能用全、用好的测试，基本没见到。出现现网问题主要原因之一。
2 要根据业务经验（业务问题、开发问题），推测常出问题的点。
3 确保用例真正被执行到位。出现现网问题主要原因之一。
4 确保用例方便执行，能准确执行
5 确保用例评审的时候，相关人员都在真正地评审，而不是拿着笔记本换个地方办公/发呆。评审时，适时互动。

bug跟踪

1 bug提交规范
1.1 标题规范（简明扼要。模块、环境、描述）
1.2 bug提交，等级严格定义。不要全是P0或者全是P4
1.3 bug详述要求有详尽步骤（1→2→3）
1.4 bug附件要全（截图、报文、log等）

2 bug日清
bug要求每天下班前全部修改完毕，只是特定bug等级、特定时间阶段可以做到日清。

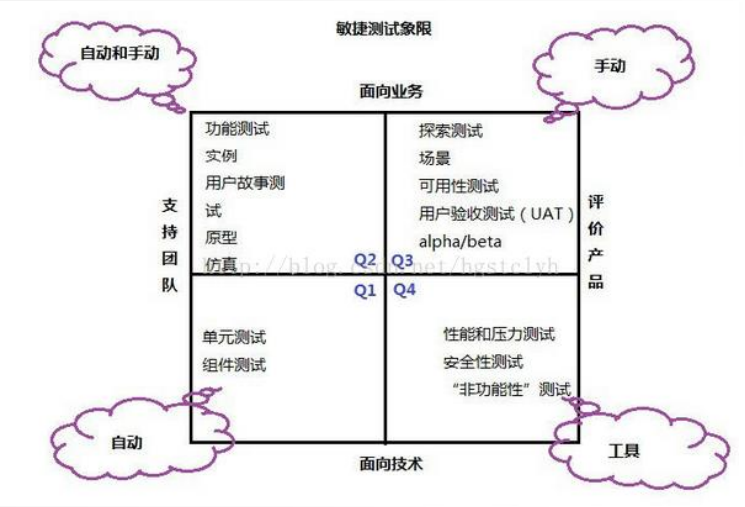
交叉测试

上线前，不同模块的测试互换测试内容，避免测试盲区。

兼容性测试

综合考虑分辨率、兼容性、硬件（芯片、内存等）、手机生产厂家等。

功能测试 3.3 敏捷测试



经典的敏捷测试4象限

敏捷是共产主义，那是我们的目标，但落实起来还有一定的距离。
而且每家公司实现敏捷的方式和程度也各有不同。
这里只讲可以从敏捷中借鉴的知识。

沟通

1.如何挖掘开发/产品没有说明或者没能力说明的需求/内容。
a.规定给出测试指导（不能轻易全回归）；b.引导说出修改内容
c.根据修改内容和经验，共同推测影响点
2.测试要测什么？为什么要测试这个？
明确需求的意义以及测试该需求的意义
3.严格约定文档/里程碑
a.文档按时更新到规定的地方。
b.重要的里程碑不能随便变更
4.定期回归会
a.定期复盘，总结经验与教训
b.TL做总结，不做引导。对事不对人
c.做好文档沉淀
d.开放式交流，挖掘反馈内容

快速迭代

1.自动化（后续介绍）
2.分批测试
尽量做到发布和提测之间，无间隔

关于全栈

不要求全栈的人，要求全栈的团队。不盲目追求全栈工程师。

性能测试 4.1

客户端稳定性提升实践

四个工作

- crash日志每日监测
- OOM /crash测试工具
- 每日持续集成
- APP源码管理

crash日志每日监测

- 阶段性（每天、每周两次、每周一次）地对crash日志进行监测、分析和归类，并且将这些统计到的crash按照优先级，以bug单的形式提给开发。
- 友盟、Fabric、自有平台
- 如果能把这些crash全部修改干净，产品的稳定性可以达到优秀的水平。但是存在延后性，也就是说APP上线之后，才能发现这些问题。

性能测试 4.1

客户端稳定性提升实践

OOM /crash测试工具

- leakcanary，可以简便可视化地监测到内存泄露的情况。因为内存泄露会积少成多，导致内存耗尽。
- 内存测工具：推荐使用阿里的易测
- 内存占用模拟工具：推荐使用腾讯的GT

每日持续集成

- 使用自动化测试工具，每天监测APP的情况
- 对某一特殊场景进行长时间操作，看是否会发生crash

APP源码管理

- 使用代码管理工具，检查代码的提交情况（规范及频率）
- 使用lint/oslint对提交代码进行静态扫描。
- 查看已测试通过的代码，是否存在代码再次提交的情况。
- 检查crash修改，是否仅仅是捕获异常（导致友盟等统计不到），还是真正修复了这个crash。

性能测试 4.2 性能测试及工具

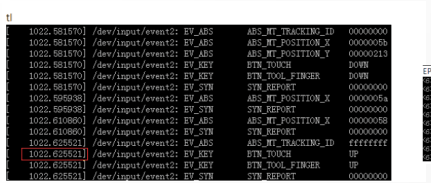
◆速度测试

1.高速摄像法
2.命令行法/SDK（可能需要或者要求的权限比较高）
3.现网端到端时延测试（自动化脚本监测法）

高速摄像法
使用高速摄像机（iPhone 6、6s、7系列自带相机的慢动作功能即可）进行录像，然后逐帧分析。方法非常简单，具体方法自行研究。

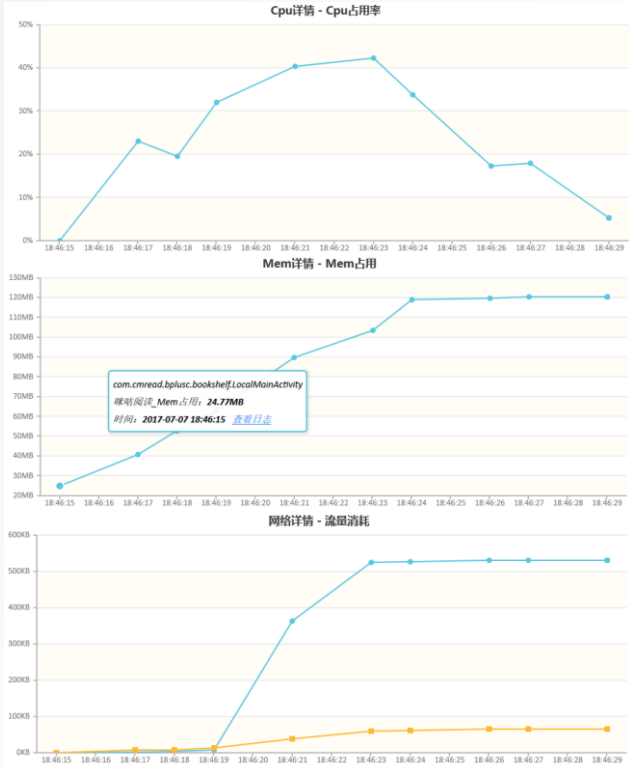
现网端到端时延测试
使用自动化脚本进行“监督”，计算时间是从某个动作（如点击button）开始到某块“图片”出来结束。
优点：最符合用户真实操作场景，可以在全国各地监控。
缺点：实现成本高，准确率略低。

命令行法/SDK
前置条件运行以下命令行
$adb shell "echo 1 > /sys/devices/soc0/soc.2/2000000.aips-bus/20f4000.epdc/mxc_epdc_debug”
$adb shell "echo 24 >/sys/devices/soc0/soc.2/2000000.aips-bus/20f4000.epdc/temperature_override”
分别在两个命令行终端下执行以下两个命令：
adb shell getevent -tl　//触摸屏幕的时间
adb shell dmesg | grep”update end marker”
//EPDC logs打印出来的时间EPDC logs上最后的时间 — 触摸屏幕的时间 = 该操作消耗的时间
如果操作联网，包含网络请求。
则网络请求消耗的时间可以通过1.开发打印日志；2.抓包 的方法得到。



性能测试 4.2 性能测试及工具

◆内存/CPU使用状况测试



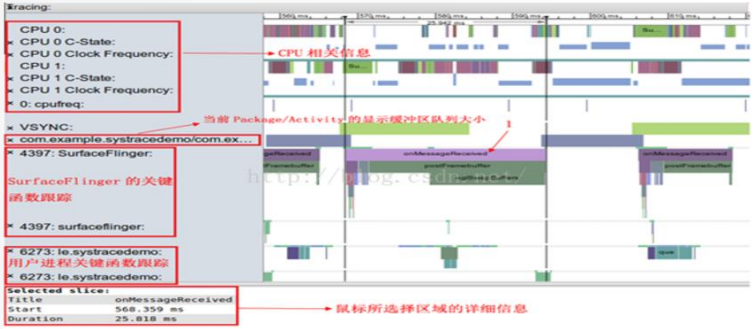
目前我们使用的是淘宝的易测
下载链接https://easytest.taobao.com/

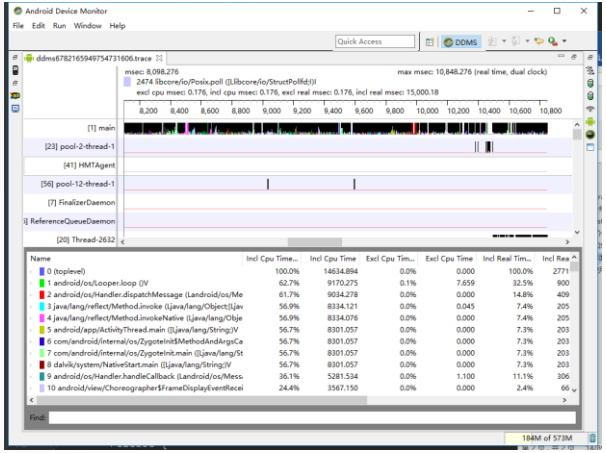
优点：
1.比较准确、稳定
2.会把资源使用情况几乎实时地显示在屏幕上。
3.上手相对容易。

其他方法：
1.Emmagee（开源）
2.腾讯的GT（开源）
3.命令行法

4.2 性能测试及工具

◆更深层次的性能测试，建议使用Systrace或者Traceview





Systrace是Android4.1中新增的性能数据采样和分析工具。它可帮助开发者收集Android关键子系统（如WindowManagerService等Framework部分关键模块、服务）的运行信息，从而帮助开发者更直观的分析系统瓶颈，改进性能。

Systrace的功能包括跟踪系统的I/O操作、内核工作队列、CPU负载以及Android各个子系统的运行状况等。在Android平台中，它主要由3部分组成：内核部分、数据采集部分、数据分析工具。

通过Systrace分析数据，可以大体上发现是否存在性能问题，但如果要知道具体情况，就需要用到Traceview这个工具。**我个人理解，两者区别一个在于从宏观层面上通过数据采集和分析诊断应用程序是否存在性能问题，当诊断存在性能问题，开发需要进一步分析具体哪些类、方法存在性能上的耗时，则需要另外一个工具从微观层面，即方法层级具体找出性能消耗主要发生在哪些地方。**

Systrace分析UI绘制时间，卡顿场景比较多，通常会结合Hierarchy Viewer来提升UI性能，虽然也可以用来发现耗时操作，但这块更

4.2 性能测试及工具

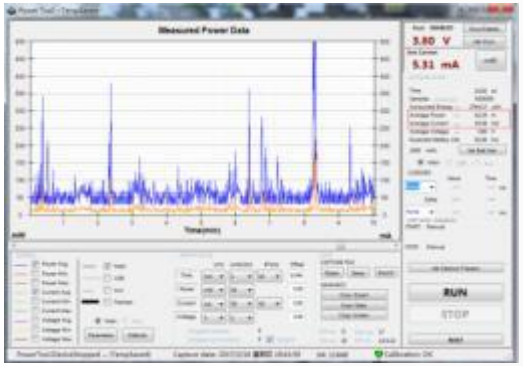
◆电量测试

使用Power monitor进行电量测试

优点：

1.精确、稳定

2.会实时地把每一步的电量消耗情况记录下来，便于分析。



其他方法：

1.安捷伦电源仪

2.命令行法

性能测试 4.2 性能测试及工具

内存泄露测试

https://www.liaohuqiu.net/cn/posts/leak-canary-read-me/

卡顿测试

https://www.tuicool.com/articles/EzYBruf

过度绘制/渲染测试

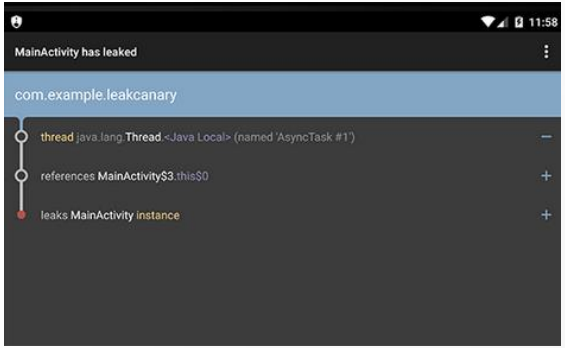
http://blog.csdn.net/a740169405/article/details/53896497

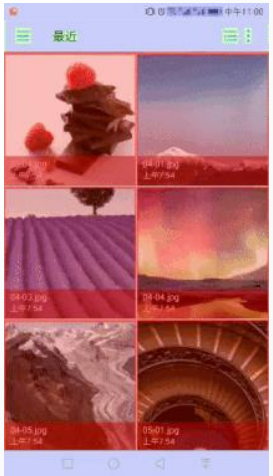
内存溢出/耗尽测试

使用腾讯的GT，可以填充内存

Doze模式测试

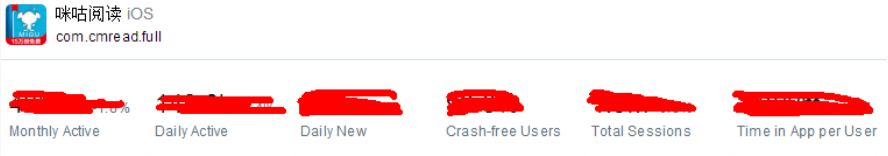
http://blog.csdn.net/manjianchao/article/details/52608626





性能测试 4.3 从测试到监控

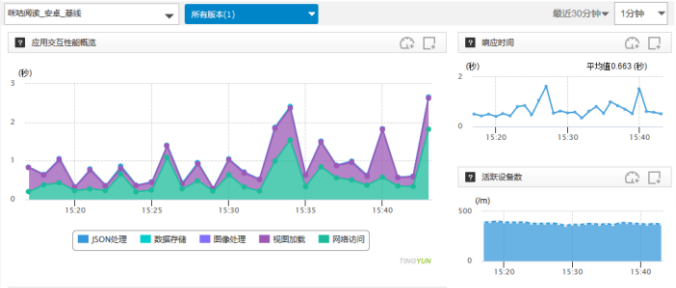
crash监控

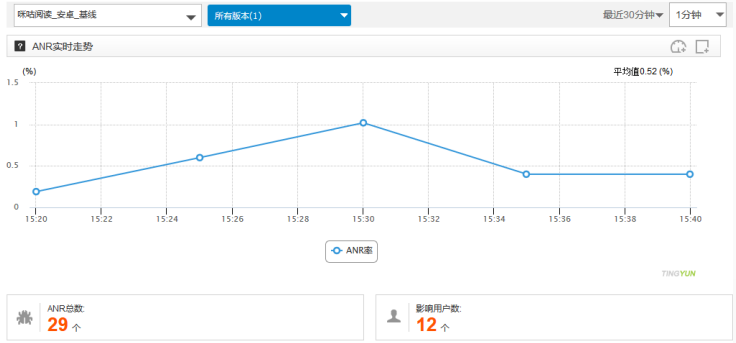


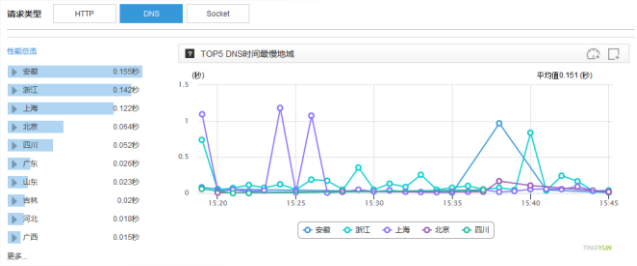
端到端时延监控

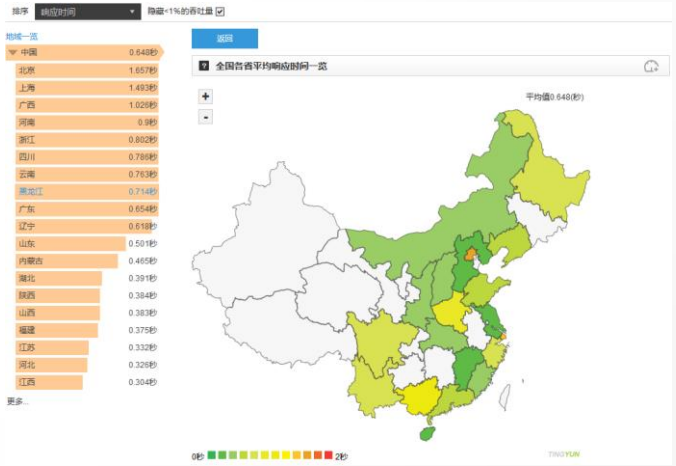
性能测试 4.3 从测试到监控

APP性能生产环境监控









自动化 5.1

UI自动化

从 自动化测试 到 测试自动化

UI自动化测试和接口自动化测试是我们公司最主要的自动化测试形式，因接口自动化测试主要由负责服务端的部门负责，我们这里重点介绍一下UI自动化测试。

目的

需要把一些重复的、耗时的工作，以自动化的方式实现，来提高测试质量和效率。

我们真正需要的是自动化测试带来的便利和结果，而不是这个研究过程。

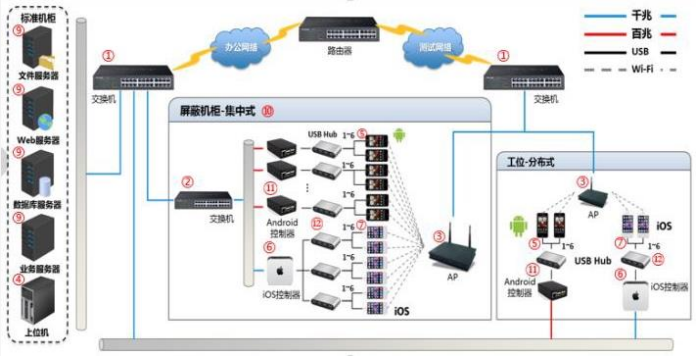
困难

覆盖率达到一定程度之后，维护成本高。

兼容性难度高，控件识别准确率低，客户端碎片化严重...

自动化 5.2 UI自动化

整个自动化的实现方式，类似于QTP，可以大大降低自动化的实现成本。



系统方案

分为私有云、客户端、机柜 3部分。

私有云 用来做数据存储和结果展示

自动化测试客户端： 主要用来编写和上传自动化脚本；

集中式屏蔽机柜：主要用来运行自动化脚本，并且防止外界环境的干扰。

5台服务器：

- 文件服务器：储存脚本、APP、测试报告
- web服务器：存储前端页面代码，渲染页面
- 数据库服务器：存储数据
- 业务服务器及上位机：负责自动化测试的业务，并进行调度

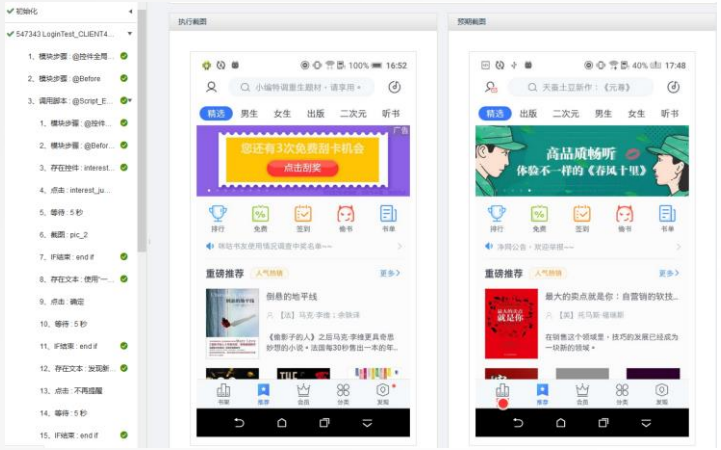
交换机： 内部数据传输

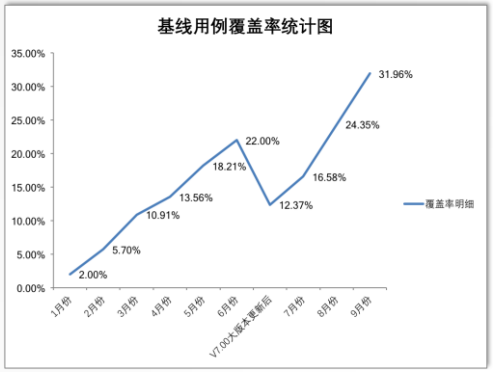
控制器：管理手机，协议转换（http转换成adb或者iproxy协议）

1台网络模拟工具： 对root的安卓手机，模拟各种网络环境，随时抓取网络报文。

AP： 外网数据路由器

自动化 5.2 UI自动化





自动化覆盖咪咕阅读、企业阅读、Kindle等所有负责的客户端。

拨测用例实现100%。

功能回归测试自动化覆盖率超过35%。

优势：

1. 纯黑盒，安全性高，操作方便。
2. 入手门槛相对较低。
3. 维护简单，覆盖率提升快。

劣势：

1. 测试工具受别人控制（其实我们用的appium等，也受控制）。
2. 需要花钱购买。

自动化 5.3

UI自动化更深层次的研究---客户端分层测试

客户端分层测试

参考源码：

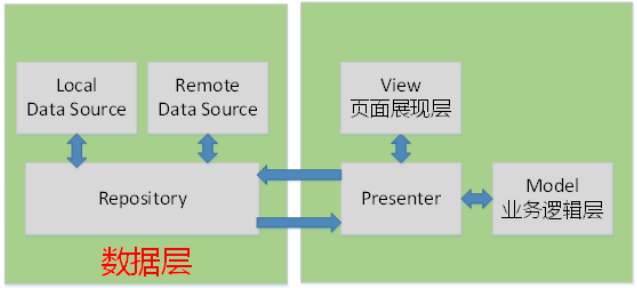
https://github.com/googlesamples/android-architecture/tree/todo-mvp/

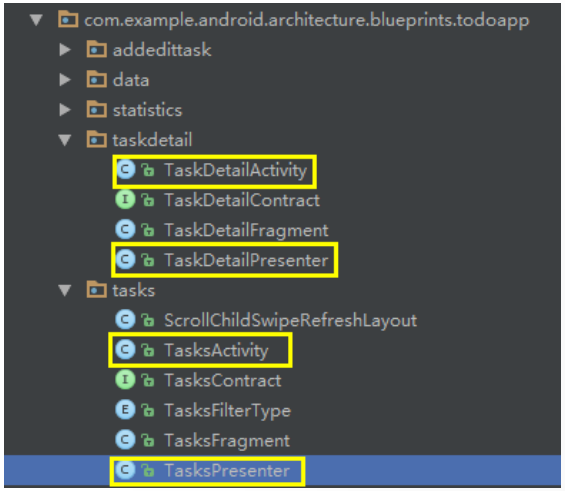
优点：精准、高效、易定位、稳定、尽早测试

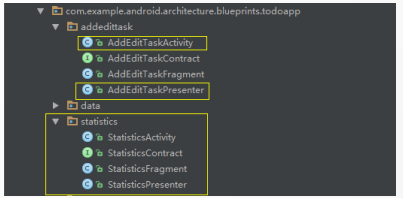
局限性：依赖于被测系统良好的分层设计、入手门槛高、无法做到端到端测试、被测程序上下文依赖太强

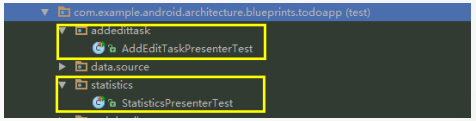
目的：规避/解决 UI自动化难以维护，覆盖率低的问题

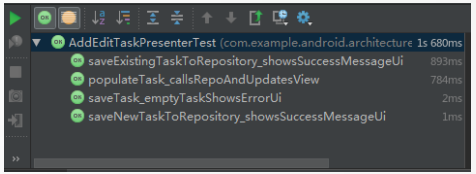
思路：不做严格分层，只将UI层和底层分开







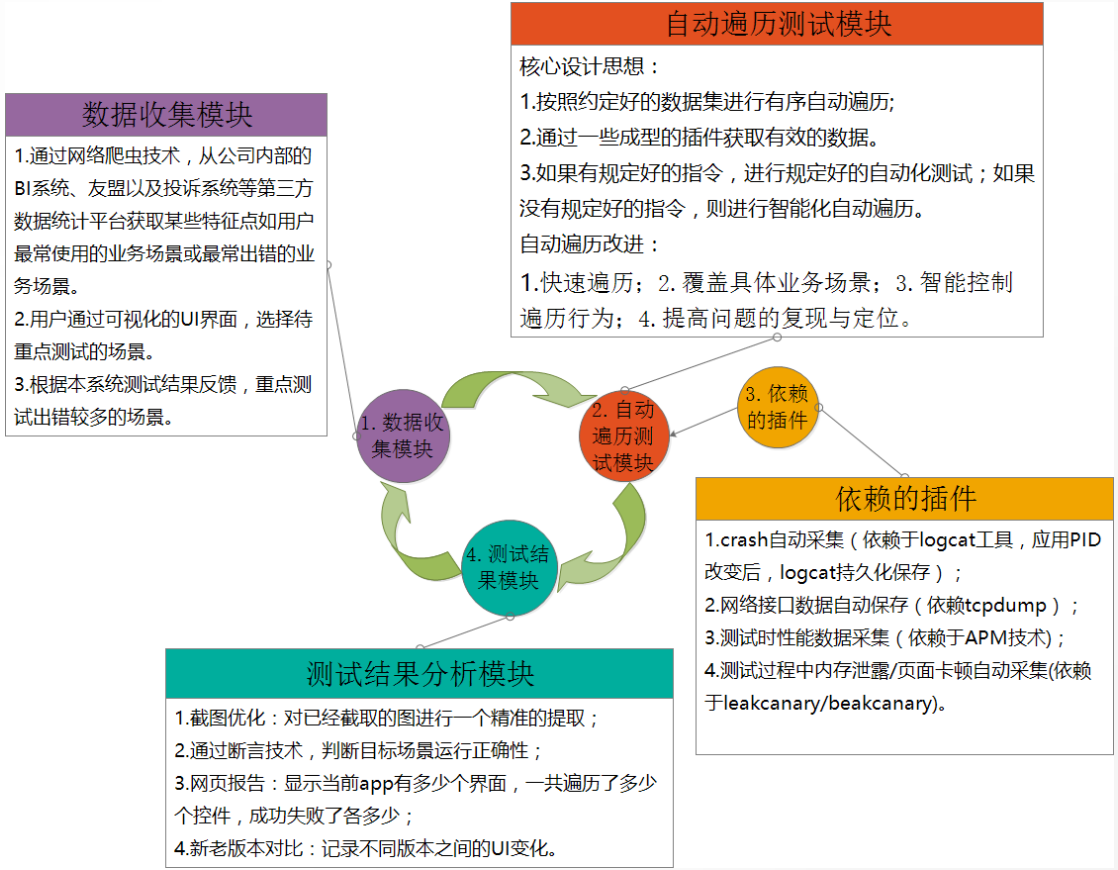




自动化 5.4 UI自动化更深层次的研究---智能自动化

智能化的UI自动化测试工具

核心思想：
1.保证自动化测试不会因测试失败而停止运行。
2.随时保存测试结果。
3.收集测试数据，并使用这些测试数据对测试进行改进。



咪咕阅读

FOR LISTENING

谢谢聆听!