TiD2016
质量竞争力大会
软件研发顶级盛会

下一代软件研发
SOFTWARE DEVELOPMENT
更高 更新 更深

在手机应用项目中建立符合持续交付的高标准自动化测试

许均扬

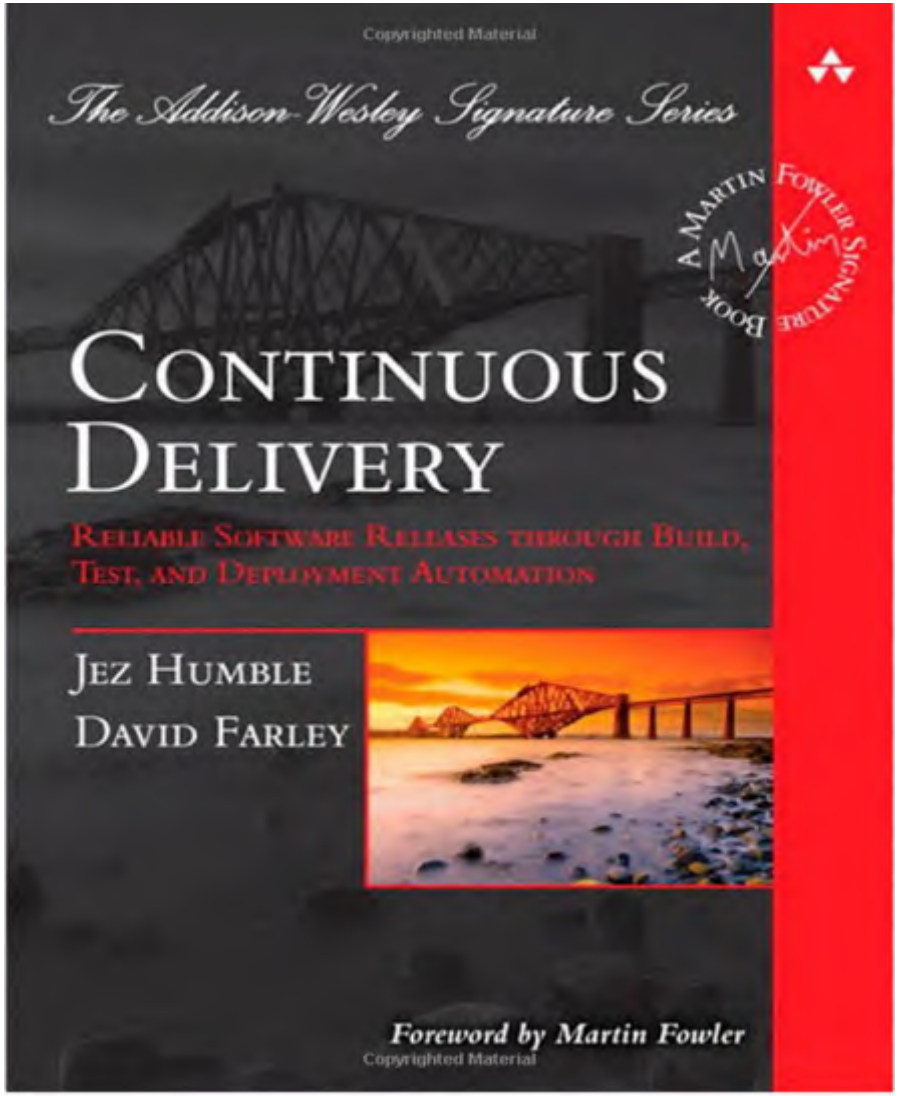
johnhsu@tw.ibm.com

Please note:

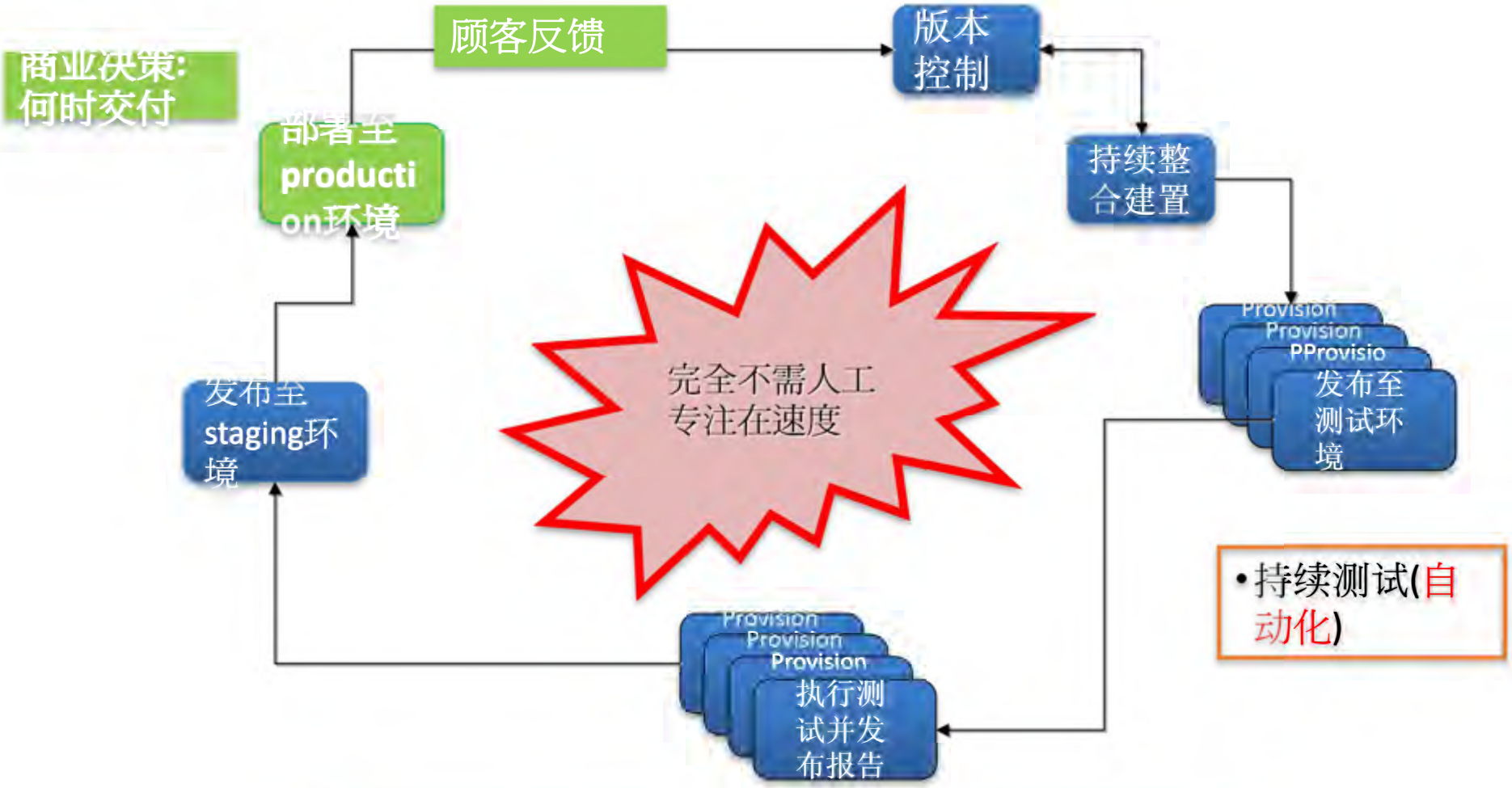
- The content in this slides only represents the view of the author, not IBM.
- The recommended technical solutions presented in this slides are not guaranteed, and the author or IBM takes no responsibility under any circumstances.

持续交付

https://www.thoughtworks.com/continuous-delivery



自动化在持续交付中的关键角色



重点是随时保持在能交付的状态

持续交付与传统自动化测试的区别

- 传统自动化:
  - 一天跑一次，甚至两三天才跑一次
  - 失败了没关系
    - 很多自动化测试甚至是已经处于每次跑必定失败的状况，长达一年以上
  - 通常不太在意代码质量跟架构，甚至有不少的团队使用录制工具建立测试
  - 不太在意测试覆盖率，低于30%是非常常见的状况
  - 跑很久无所谓，因为一天才跑一次

持续交付与传统自动化测试的区别

- 持续交付自动化
  - 一天跑十次甚至几十次
  - 不允许非bug造成的失败
  - 需要高覆盖率，80%以上是基本要求
  - 需要以对待产品代码级别的标准来要求代码质量与架构
  - 需要在短时间内结束
  - 真的失败需要在短时间内快速找出问题点
    - 测试报告的完整度与质量非常重要

持续交付自动化的挑战

- 够快
- 非常稳定
- 失败时能快速精準找出问题

要多快?

- Build + Unit Test不能超过15分钟，最好在十分钟以内
- Acceptance Test最好在30分钟以内，最多不超过60~90分钟

要多稳定?

- 1%失败率很稳定吗?
  - 假设每一个test case有1%概率由非bug问题造成失败
  - 100个test case的test suite跑一次完全没失败的概率是0.99^100 = 36%
  - 1000个test case呢?几乎每次都至少会有一个test case失败
- 持续交付要求的稳定是”完全不会因为非bug问题失败”
  - 事实上很难做到
  - 但至少每一个test case非bug造成失败率不能大于百万分之一

要能多快找出问题?

- 假设一天八小时内跑十次，每次跑40分钟
- 每次跑的间隔平均是8*60/10 = 48分钟
- 所以如果失败，只有8分钟在下一次code check in之前找出问题
- 或是一旦失败就锁死source control不让code check in，直到问题解决或是roll back
  - 无论采取哪一种方法，找出问题都是分秒必争

有可能做到吗?

- 够快
- 非常稳定
- 失败时能快速精准找出问题

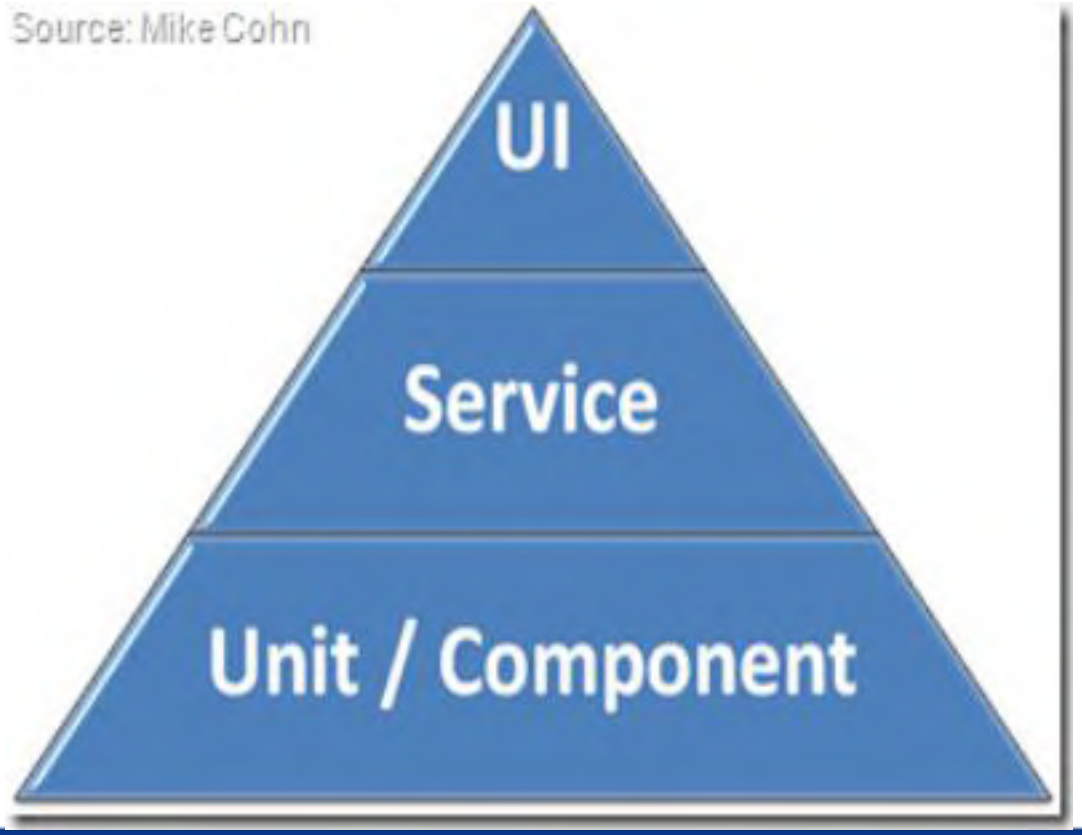
光是做到一项就是很大挑战，何况是全部

解决方法

- 挑战显然是巨大的，因此解决方法也不可能只采用单一手段
- 战略面跟战术面都必须要有各种方法，多管齐下

战略面

- 最重要第一点，自动化测试策略的制定
- 战略面一开始就错了，战术面再怎么强还是全盘皆输



战略面

- 组织，流程与文化
- 开发团队写自动化测试
  - 很多公司直接要求现有的手动测试团队负责开发自动化测试
    - 自动化测试质量是否高，跟产品代码的可测试性关联也很高
    - 新的story无法马上就有自动化测试，而是要等几星期到几个月
    - 术业有专攻,写好代码不是几星期就能会的事
    - Whole team approach
  - 测试团队负责手动测试，测试案例review与工具平台开发维护
  - 以对待production代码的态度对待自动化测试代码

战略面

- 流程上必须要确保pipeline纪律的严格执行
  - 就算有全世界最好的自动化测试，没人看没人用的话也只是废物
  - 测试失败必须立刻解决←非常重要
- 文化
  - 持续改进的文化
  - 维持持续交付pipeline的决心

战略面: Behavior-Driven Development

- Project Manager与其他主管最常问的问题:我们自动化测试有哪些，覆盖的哪些scenarios?
  - 请他们去看代码?行不通
  - 解决方案: Automation as document

```
Feature: As a user, I should be able to send and reply a simple mail to several recipients

Scenario: Composed mail from UI
Given I start the mail client
And I compose a new mail "Test Simple Mail" at device
When I send the mail
Then The mail "Test Simple Mail" that I just sent should be received by its recipients
```

战略面

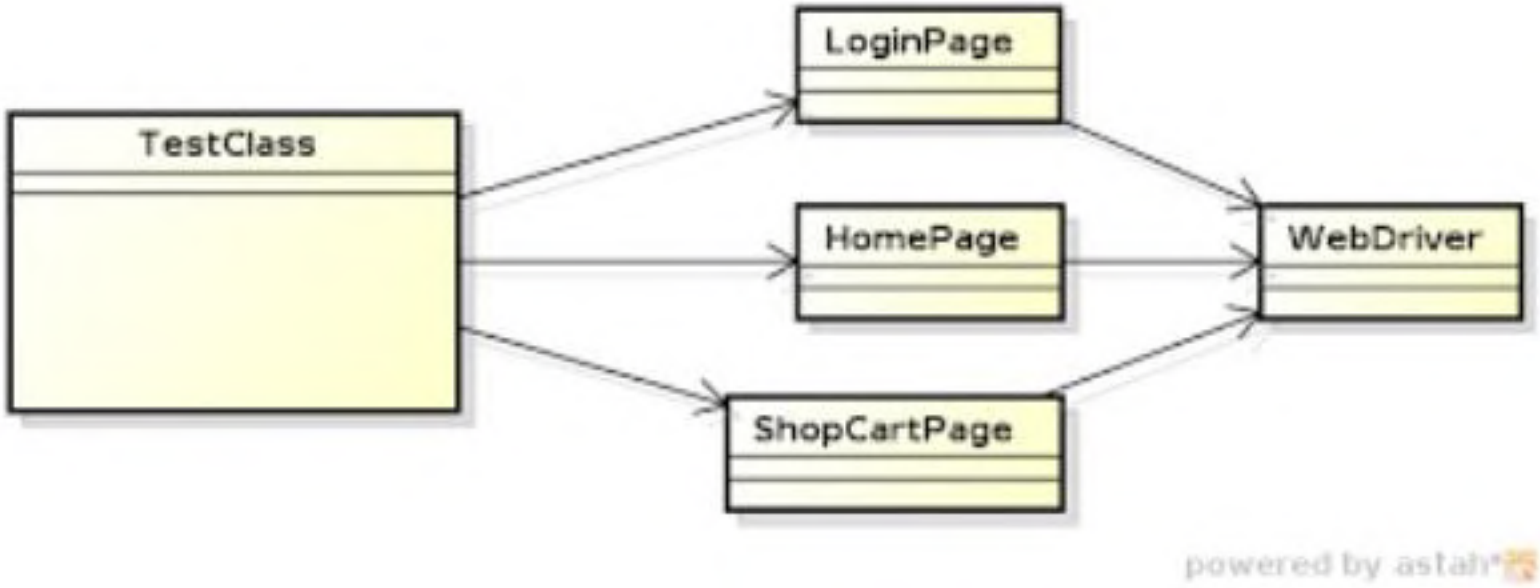
- 工欲善其事，必先利其器
- 年年都有新的测试工具，因此要时时更新知识
- 测试工具最好要能
  - 自动处理UI的延迟，让测试代码中不需要写sleep
  - 简单易懂
  - 够稳定
  - 够快
  - 官方支持优先，其次才考虑开源第三方软件
    - 当官方版很差时，没别的选择只能用开源第三方软件
    - 但一旦官方版达到一定水平以上，开源软件就可以功成身退了
    - Android就用Espresso，iOS就用Xcode UI Test
    - Espresso出来，Robotium就再见了。Xcode UI Test出来，KIF就被淘汰了。

战略面:写Test Case的大原则

- 非到万不得已，绝对不写sleep
  - Sleep是不稳定测试的最大来源
  - Callback->Periodic Poll->Sleep
- 每个Test Case必须能独立执行，不依赖其他Test Case帮它设定环境，也不影响其它Test Case
- 避免过度复杂的测试代码
  - 虽然说要用产品代码级别要求代码质量，但还是稍有不同，如何拿捏很重要
  - 避免复杂类别架构，if/else与switch
- 使用view id或是accessibility label/id定位view
  - 字符串会变
  - 坐标更容易变

战略面: Page Object Pattern

- Proposed by Martin Fowler
- Picture from Roger Almeida's blog:http://roger-almeida.blogspot.tw/2012/01/page-object-pattern-webdriver-spring.html



战略面:限制测试Scope

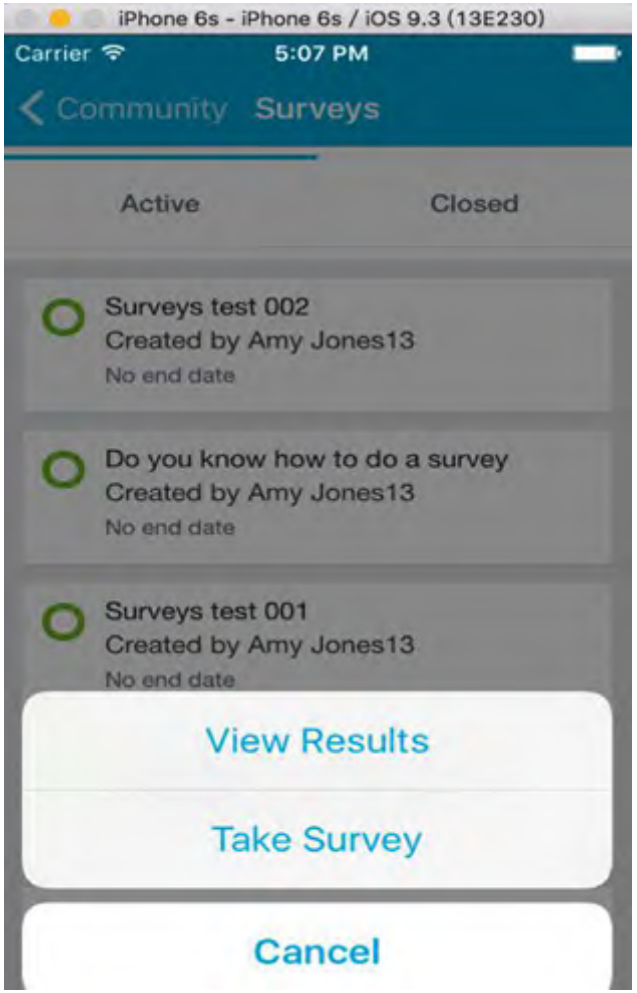
- Unit Test只应该测试单一method
  - 不连网
  - 不连数据库
  - 不调用任何硬件功能如GPS，震动等等
  - Mock第三方软件或是OS API的调用
- Acceptance test应该只测试单一component
  - 不是integration test
  - Client或是Server，而不是end to end
  - Mock client或mock server
  - 或是把client跟server放在同一台机器上

战术面:避免Runtime决定测试路径

- 怎么处理这问题?
  - If(waitfor(menu exists))
    { click(“View Results”); }
- 这样做对吗?
- 我们是不是忘了一件事?
  - 在写测试案例时,我们肯定知道这时候它会走哪条路
  - 既然如此何必runtime判断
- takeSurvey(String surveyName, boolean menuWillPopup) {

  .....

  If(menuWillPopup)
  { click(“View Results”); }
}



战术面: Incremental Periodic Poll

```
private static long[] waitPeriods = { 10, 30, 50,
    100, 200, 300, 500,
    TimeUnit.SECONDS.toMillis(1),
     TimeUnit.SECONDS.toMillis(2),
     TimeUnit.SECONDS.toMillis(3),
     TimeUnit.SECONDS.toMillis(4),
     TimeUnit.SECONDS.toMillis(5),
     TimeUnit.SECONDS.toMillis(10),
     TimeUnit.SECONDS.toMillis(15),
     };
```

```
public interface TestAfterWait {
    public boolean test();
};
```

```
public static boolean waitFor(TestAfterWait testAfterWait, long timeout)  {
    long startTime = System.currentTimeMillis();
    for (int i = 0; i < waitPeriods.length; i++)
    {
       try
       {
          Thread.sleep(waitPeriods[i]);
       }
       catch (InterruptedException e)
       {
          e.printStackTrace();
       }
       if (testAfterWait.test())
       {
          return true;
       }
       if((System.currentTimeMillis() - startTime) > timeout) {
           return false;
       }
    }

    return false;
}
```

战术面:即使是UI测试，还是要尽量减少UI动作

- 假设以下测试案例，测试目标为一cloud同步服务
  - 上传一文档 -> 编辑文档描述 -> 将文档与另一人共享 -> 上传同一文档的更新 -> 删除文档
- 测试案例2
  - 上传一文档 -> 在文档上留言 -> 将文档公开给所有人看 -> 删除文档
- 同样的动作，只需要在UI测试一次就够了
  - 第二个以后的测试案例，测试过的步骤都透过API执行

战术面:跳过已经被其他测试案例测试过的步骤

- 假设以下测试步骤,目的为测试购物网站是否能正确使用信用卡付款
  - 登入在线购物网站 -> 寻找商品 -> 找到商品,确认数量,放入购物车,结账 -> 输入地址电话 -> 选择以信用卡付款 -> 付款 -> 确认付款完成
- 同样的购物网站可能有数百个测试案例,前面几步都完全一样
  - 手动测试没有别的方法准备测试数据
  - 但自动化测试中,我们有别的选择
    - 用API或是直接call production code产生我们要的测试数据

战术面:对付工具的不稳定

- 测试工具也是人写的软件，也会有bug
  - 等测试工具没问题才开始自动化测试?那永远都不会开始
- 手机上常见的测试工具bug
  - 点不准
  - 点了没作用
  - 动画导致时间差
  - 长按变短按,短按变长按
  - 滑动失败
- 解决方法
  - 重试
  - 还原后重试
  - 关掉动画

重试测试步骤

- Espresso的重试

```
onData(allOf(is(instanceOf(Email.class)), withItemContent(subject)))
    .perform(click(pressBack()));
```

- 自己写的重试

```
boolean result = testUtils.waitForViewToBecomeVisible(mailDetailId);
if (!result)
{
    pressBack();
    onData(allOf(is(instanceOf(Email.class)), withItemContent(subject)))
            .perform(click(pressBack()));
```

重试测试步骤

- 或是直接将这些代码包装成一个generic method

```
public boolean clickOnView(String viewId, ClickSuccessEvaluator evaluator, LongClickRollback rollback
                           , int retryCounts) {
   onView(withId(viewId)).perform(click(rollback.rollback()));
   for(int i = 0; i < retryCounts; i++) {
      if(evaluator.evaluate()) {
         return true;
      }
      onView(withId(viewId)).perform(click(rollback.rollback()));
   }

   return evaluator.evaluate();
}
```

战术面:详细的测试报告

- 测试失败时只有几分钟时间找出问题
- 报告应该要有
  - BDD中每一步的描述
  - 每一步之下细项动作的记录
  - 更细的UI lifecycle记录,如Espresso中的activity lifecycle
    如果是UI测试:失败测试的屏幕截图,最好有完整的video replay
- 报告要能分component列表,让人一眼就看出失败出在哪个component
- 常见的反应
  - 测试能跑就好,为什么要我写一堆Log?
  - 为什么还要规定Log的规范,会不会太麻烦?

持续改进

- 跟所有Agile practice一样,持续交付的自动化测试永远可以有改进空间
- 如同对待production代码一样的对待自动化测试代码,持续改进它,才是最重要的
- 只有使用技术跟工具，而没有将持续改进的文化深植到团队中，只能事倍功半

•Statement of Good Security Practices: IT system security involves protecting systems and information through prevention, detection and response to improper access from within and outside your enterprise. Improper access can result in information being altered, destroyed, misappropriated or misused or can result in damage to or misuse of your systems, including for use in attacks on others. No IT system or product should be considered completely secure and no single product, service or security measure can be completely effective in preventing improper use or access. IBM systems, products and services are designed to be part of a lawful, comprehensive security approach, which will necessarily involve additional operational procedures, and may require other systems, products or services to be most effective. IBM does not warrant that any systems, products or services are immune from, or will make your enterprise immune from, the malicious or illegal conduct of any party.

Q&A