

《王国纪元》 自动化测试与持续集成

- 《王国纪元》 自动化测试与持续集成
 - 摘要
 - 分享的初衷
 - 如何进行游戏自动化测试
 - 什么是游戏自动化测试?
 - 游戏自动化思路
 - 图像识别-OPENCV
 - 什么是opencv
 - opencv的安装
 - 测试opencv
 - 示例1: 识别IGG Logo
 - 示例2: 通过摄像头捕获'IGG'
 - 设备控制 - ADB
 - adb是什么
 - adb的安装
 - 测试adb
 - 示例1: 通过adb打开《王国纪元》应用
 - 开始我们的第一个测试用例
 - 步骤1: 获取截图
 - 步骤2: 图像识别
 - 步骤3: 按键模拟
 - 自动化测试的成本与收益
 - 自动化测试成本很高
 - 如何最大化收益
 - 持续集成与自动化测试相结合
 - 持续集成的目的
 - 传统瀑布存在的问题
 - 船小好调头(小版本迭代)
 - 持续集成与自动化测试的关系
 - 持续集成解决研发的噩梦
 - 持续集成为什么要引入自动化测试
 - Jenkins做持续集成
 - Jenkins使用场景
 - 将自动化测试集成到jenkins任务中
 - 将测试的结果以报告形式反馈用户

摘要

主题：《王国纪元》自动化测试与持续集成

摘要：借助opencv图像识别与**android adb**实现游戏自动化测试，思考能否通过自动化测试达到节省人力、缩短**bug**发现周期、提高效率等目的？

分享的初衷

遇到了一些问题：

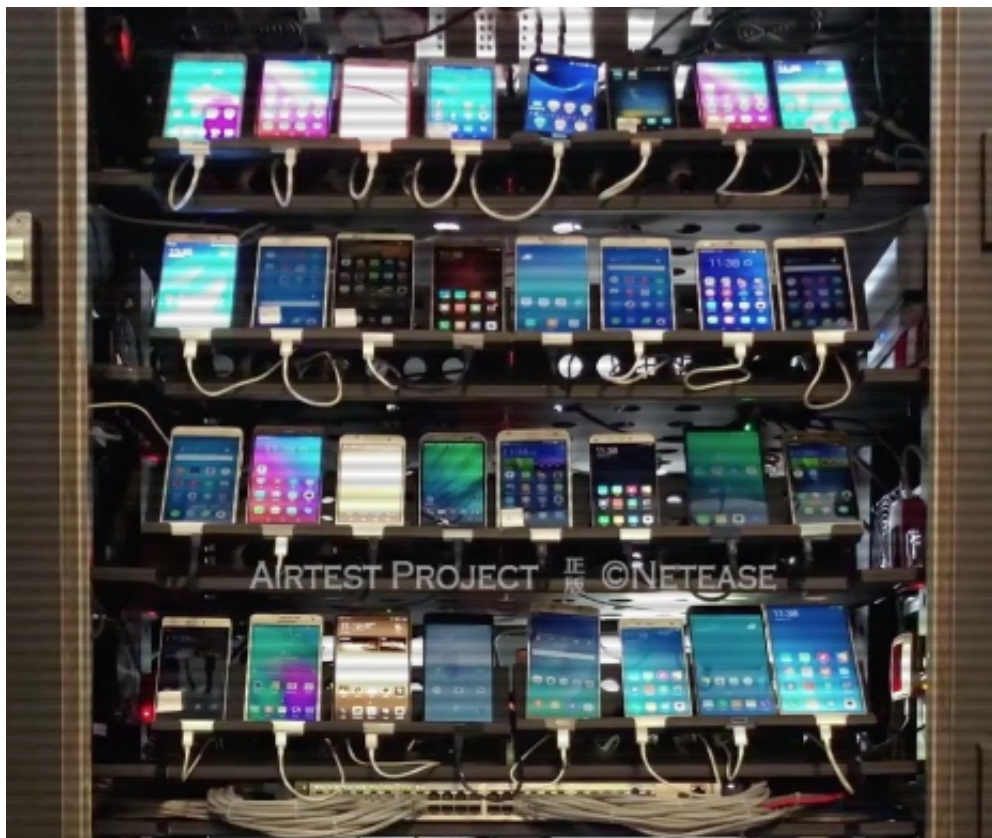
- 明明XX看起来是正常的，却有人反馈用户登陆不了；
- 我只是做了一个小调整，不确定这个改动会不会影响到业务；
- 研发新提了一个小修改测试包，其他功能是否应该进行回归测试；
- 调整数值为了测试某个场景，但却重复进行无价值的新手教程；

思考问题背后的问题：

- 如何缩短BUG发现的周期？
- 如何及时反馈用户的真实状况？
- 如何提高研发、测试效率？

其他游戏公司如何解决这些问题：

在网易游戏内部，持续集成被应用在梦幻西游、大话西游、阴阳师、荒野行动等数十个游戏的自动化测试中。在游戏放出前，通常会在数百台安卓手机上测试游戏的兼容性。



如何进行游戏自动化测试

什么是游戏自动化测试？

其实我们平时也有接触自动化测试，比如单元测试、接口测试、业务监控等都属于自动化测试的一种。但是手游一般使用引擎开发（cocos2d、unity3d..），虽然在开发过程中也有按钮等控件的概念，但当运行时由引擎渲染后就变成了一副简单的图片，因此难以定位控件。为了解决定位控件的难题，我们要借助opencv图像识别和android adb来实现游戏自动化测试过程。

游戏自动化思路

图像定位测试的核心思路其实是利用adb来操作设备，用opencv实现图像区域匹配，匹配成功后计算目标位置然后触发adb。

简单来说就是如下步骤：

截图 -> 图像识别 -> 计算位置 -> 点击位置



从设备截图

1

通过ADB操作设备，屏幕截图后将图片拉到本地。

区域图像识别

2

借助opencv在截图中识别目标图像，并算出图像位置。

模拟设备输入

3

通知ADB操作设备，模拟触摸动作。

图像识别-OPENCV

什么是opencv

百度百科：OpenCV是一个基于BSD许可（开源）发行的跨平台计算机视觉库，可以运行在Linux、Windows、Android和Mac OS操作系统上。它轻量级而且高效——由一系列 C 函数和少量 C++ 类构成，同时提供了Python、Ruby、MATLAB等语言的接口，实现了图像处理 and 计算机视觉方面的很多通用算法。

opencv的安装

```
sudo pip install opencv-python
```

测试opencv

选一张图片拷贝以下代码，用opencv打开测试是否安装成功

```
#!/usr/bin/env python
# coding=utf-8

import cv2 as cv

img = cv.imread("test.jpg")
cv.namedWindow("Image")
cv.imshow("Image",img)
```

```
cv.waitKey(0)
cv2.destroyAllWindows()
```

示例1：识别IGG Logo

函数格式

模板识别'IGG'图像

```
#!/usr/bin/env python
# coding=utf-8

"""图像匹配"""

__author__ = 'xiaocai'

import cv2

# 读取图像
targetImg = cv2.imread("test.jpg")
findImg = cv2.imread("cut03.jpg")

# 模板尺寸
findHeight, findWidth, findChannel = findImg.shape[::]

# 模板匹配
result = cv2.matchTemplate(targetImg, findImg, cv2.TM_CCOEFF_NORMED)
minVal, maxVal, minLoc, maxLoc = cv2.minMaxLoc(result)

# 画点
cv2.circle(targetImg, maxLoc, 3, (0, 255, 255), -1)

# 矩形选中匹配模板
cv2.rectangle(targetImg, maxLoc, (maxLoc[0]+findWidth, maxLoc[1]+findHeight),
, (0,255,0), 1)

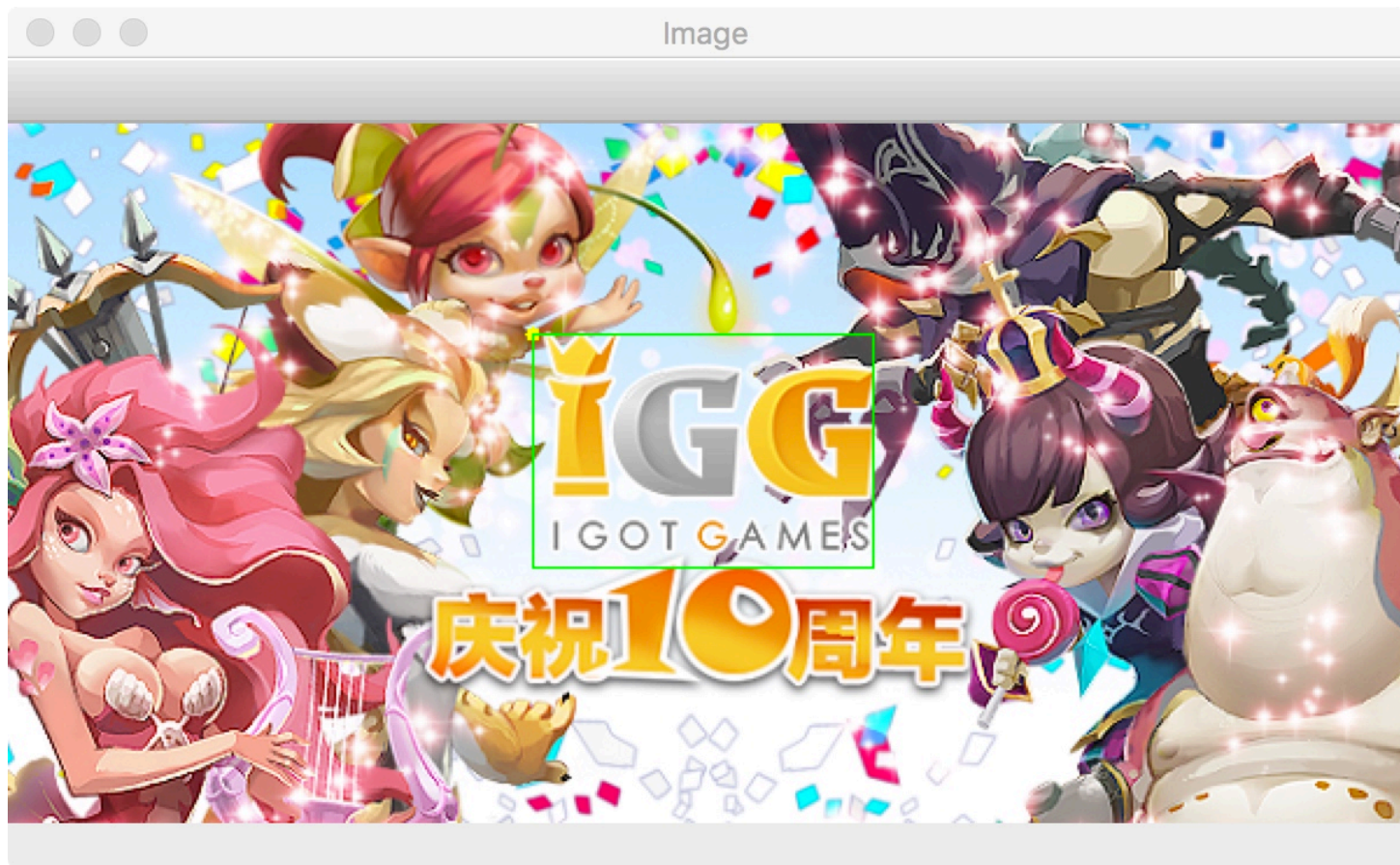
# 打印图片
cv2.namedWindow("Image")
cv2.imshow("Image",targetImg)
cv2.waitKey(0)
cv2.destroyAllWindows()
```




igg_bg.png



igg_icon.png



示例2: 通过摄像头捕获'IGG'

```
#!/usr/bin/env python
# coding=utf-8

"""捕获视频,识别图像"""

__author__ = 'xiaocai'

import cv2

# 匹配图片
findImg = cv2.imread("igglog.png")
findHeight, findWidth, findChannel = findImg.shape[::]
```

```
# 指定摄像头, 设置视频尺寸
cap = cv2.VideoCapture(0)
cap.set(3, 640)
cap.set(4, 480)

while(True):
    # 读取每一帧
    ret, frame = cap.read()

    # 模板匹配
    result = cv2.matchTemplate(frame, findImg, cv2.TM_CCOEFF_NORMED)
    minVal, maxVal, minLoc, maxLoc = cv2.minMaxLoc(result)

    # 相似度>0.5时画框
    if maxLoc > 0.5 :
        cv2.rectangle(frame, maxLoc, (maxLoc[0]+findWidth, maxLoc[1]+findHeight), (0,255,0), 1)

    # 窗口显示当前帧
    cv2.imshow('frame', frame)

    # 按q键退出
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

GIF



设备控制 - ADB

adb是什么

ADB，即 Android Debug Bridge(调试桥)，它是 Android 开发/测试人员不可替代的Debug工具，该工具可以帮助你管理设备或模拟器 的状态。

adb的安装

windows系统

也可以通过下载android SDK的方式手动安装这里不做说明，下载地址：

Win:

在 <http://www.android-studio.org/> 下载命令行工具

```
wget https://dl.google.com/android/repository/sdk-tools-windows-3859397.zip
```


MAC:

通过Homebrew命令安装

```
brew cask install android-platform-tools
```

测试adb

连上手机的数据线或打开模拟器，查看能不能找到设备

```
(venv) $ demo5_game adb version
Android Debug Bridge version 1.0.32
(venv) $ demo5_game adb devices
List of devices attached
192.168.56.100:5555 device
```

示例1：通过adb打开《王国纪元》应用

命令格式：

```
adb shell am start [options] <INTENT>
```

调起应用：

通过logcat找到Activity

```
adb shell logcat | grep com.igg.android.lordsmobile
```

```
D/dalvikvm( 4480): Shared lib '/data/app-lib/com.igg.android.lordsmobile-1/libBattleSimDll.so' already loaded in same CL 0xa65f20e8
D/dalvikvm( 4480): Trying to load lib /data/app-lib/com.igg.android.lordsmobile-1/libBattleSimDll.so 0xa65f20e8
D/dalvikvm( 4480): Shared lib '/data/app-lib/com.igg.android.lordsmobile-1/libBattleSimDll.so' already loaded in same CL 0xa65f20e8
D/dalvikvm( 4480): Trying to load lib /data/app-lib/com.igg.android.lordsmobile-1/libBattleSimDll.so 0xa65f20e8
D/dalvikvm( 4480): Shared lib '/data/app-lib/com.igg.android.lordsmobile-1/libBattleSimDll.so' already loaded in same CL 0xa65f20e8
D/FacebookSDK.com.facebook.appevents.AppEventQueue( 4480): Params: {"application_package_name":"com.igg.android.lordsmobile","advertiser_tracking_enabled":true,"event":"CUSTOM_APP_EVENTS","advertiser_id":"3baa2d7a-ee1e-454a-8075-df65095f8977","installer_package":"com.android.vending"}
D/FacebookSDK.com.facebook.appevents.AppEventQueue( 4480): Params: {"application_package_name":"com.igg.android.lordsmobile","advertiser_tracking_enabled":true,"event":"CUSTOM_APP_EVENTS","advertiser_id":"3baa2d7a-ee1e-454a-8075-df65095f8977","installer_package":"com.android.vending"}
I/ActivityManager( 2033): Killing 4480: com.igg.android.lordsmobile/u0a10070: remove task
W/InputDispatcher( 2033): channel 'a6b6cee8 com.igg.android.lordsmobile/com.igg.iggSDKbusiness.IGGSDKPlugin (server)' ~ Consumer closed input channel or an
E/InputDispatcher( 2033): channel 'a6b6cee8 com.igg.android.lordsmobile/com.igg.iggSDKbusiness.IGGSDKPlugin (server)' ~ Channel is unrecoverably broken and
W/InputDispatcher( 2033): Attempted to unregister already unregistered input channel 'a6b6cee8 com.igg.android.lordsmobile/com.igg.iggSDKbusiness.IGGSDKPlu
I/WindowState( 2033): WIN DEATH: Window{a6b6cee8 u0 com.igg.android.lordsmobile/com.igg.iggSDKbusiness.IGGSDKPlugin}
```

启动app

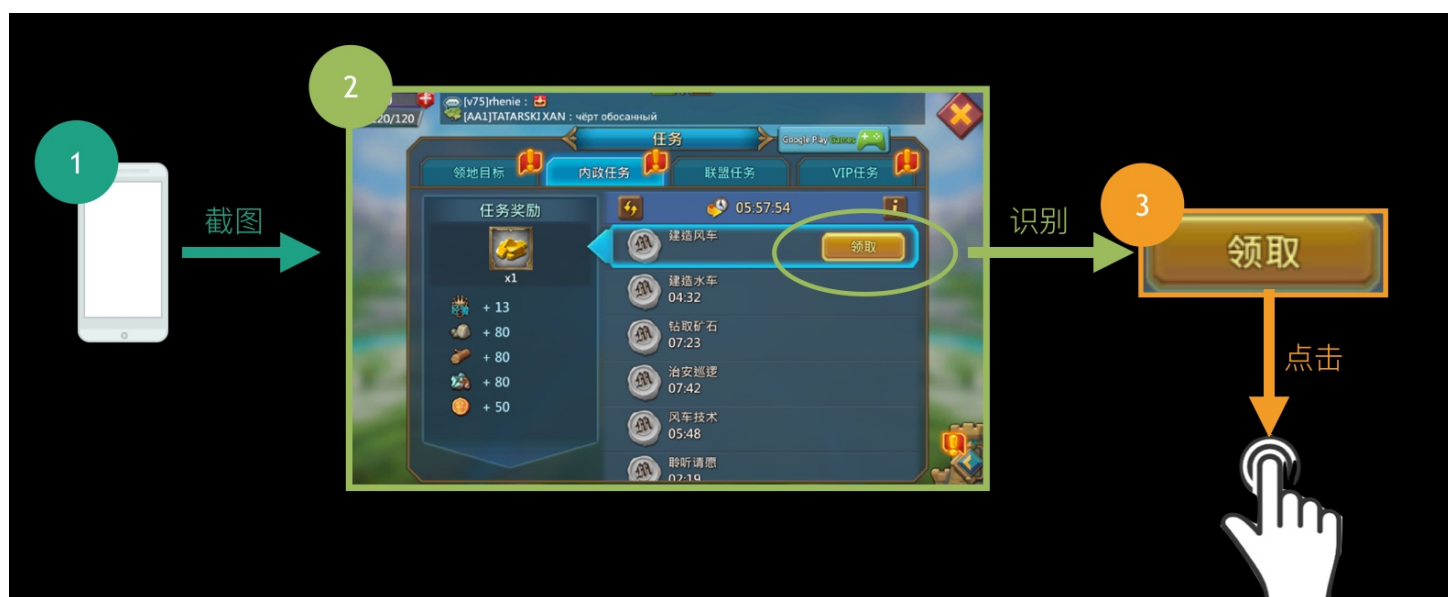
```
adb shell am start -n com.igg.android.lordsmobile/com.igg.iggsdkbusiness.IGG
SDKPlugin
```

开始我们的第一个测试用例

图像定位测试的核心思路其实是利用adb来操作设备，用opencv实现图像区域匹配，匹配成功后计算目标位置然后触发adb。

简单来说就是如下步骤：

截图 -> 图像识别 -> 计算位置 -> 点击位置



步骤1：获取截图

截图我们需要使用到adb的 `screencap` 和 `pull` 命令，先通过 `screencap` 命令将截图保存到设备里，在通过 `pull` 命令将截图文件导出到电脑。

```
adb shell screencap -p /sdcard/sc.png
adb pull /sdcard/sc.png
```

在python中执行

```
import commands
commands.getstatusoutput('adb shell screencap -p /sdcard/sc.png')
commands.getstatusoutput('adb pull /sdcard/sc.png')
```

步骤2：图像识别

前面已经介绍了 `cv2.matchTemplate` 函数的使用，这里我们需要做个调整，将上个列子中的 `targetImg` 换成设备截图，`findImg` 换成'X'关闭按钮。

```
import cv2, commands

# 设备截图
commands.getstatusoutput('adb shell screencap -p /sdcard/sc.png')
commands.getstatusoutput('adb pull /sdcard/sc.png')

# 读取图像
targetImg = cv2.imread("sc.png")
findImg = cv2.imread("close_btn.jpg")

# 模板尺寸
findHeight, findWidth, findChannel = findImg.shape[::]

# 模板匹配
result = cv2.matchTemplate(targetImg, findImg, cv2.TM_CCOEFF_NORMED)
minVal, maxVal, minLoc, maxLoc = cv2.minMaxLoc(result)

print maxLoc
```

步骤3：按键模拟

`maxLoc` 返回了 `close_btn.jpg` 图标在截图中的左上角位置，而我们需要点击中间的位置。

```
# 获取close_btn.jpg图像尺寸
findImg = cv2.imread("close_btn.jpg")
findHeight, findWidth, findChannel = findImg.shape[::]

# 计算中间位置
pointCentre = (maxLoc[0]+(findWidth/2), maxLoc[1]+(findHeight/2))
print pointCentre
```

最后一步，通过 `input tap <x> <y>` 命令来模拟按键行为

```
shell input tap 500 300
```

自动化测试的成本与收益

自动化测试成本很高

从上面的演示中大家可能会认为引入自动化测试的成本太大了，并且需要招聘编码能力不亚于研发的测试人员，反而没有手工测试来的高效廉价。

实际上目前已经有成熟的自动化测试工具，即使测试人员不会编码也能够借助辅助测试手段来完成自动化测试，甚至能够通过填表格、截图..等方式来完成自动化测试。

自动化测试工具：

- airtest
- openAtx
- itestin
- robot framework
- TestWriter
- ...

如何最大化收益

什么项目适合自动化：

回归测试为主的项目，即需要长期做支持维护的产品。或者有过去版本需要长期做支持维护的产品。对于产品的UI部分如果会频繁改动，可以做比较低的自动化。对于接口比较稳定的服务组件可以提高自动化率。

自动化的介入时间点：

一个项目的初期可能不太适合自动化。因为项目初期用户界面和接口没有稳定，自动化代码会被动的被要求频繁改变，维护成本非常高。自动化收益不好。而反而手动测试能够快速发现问题，反馈给开发人员。而到了项目后期和维护期，自动化再介入为回归测试做准备，可以最大化自动化收益。

自动化测试无法替代手工测试：

自动化测试不可能去代替人工测试，自动化测试时是为了解决特定场景下的测试主要目的，是为了降低手工的重复劳动。需求只是需求，因为很多测试用例是无法被自动化的，或者自动化代价太高。

选择合适的工具与方案：

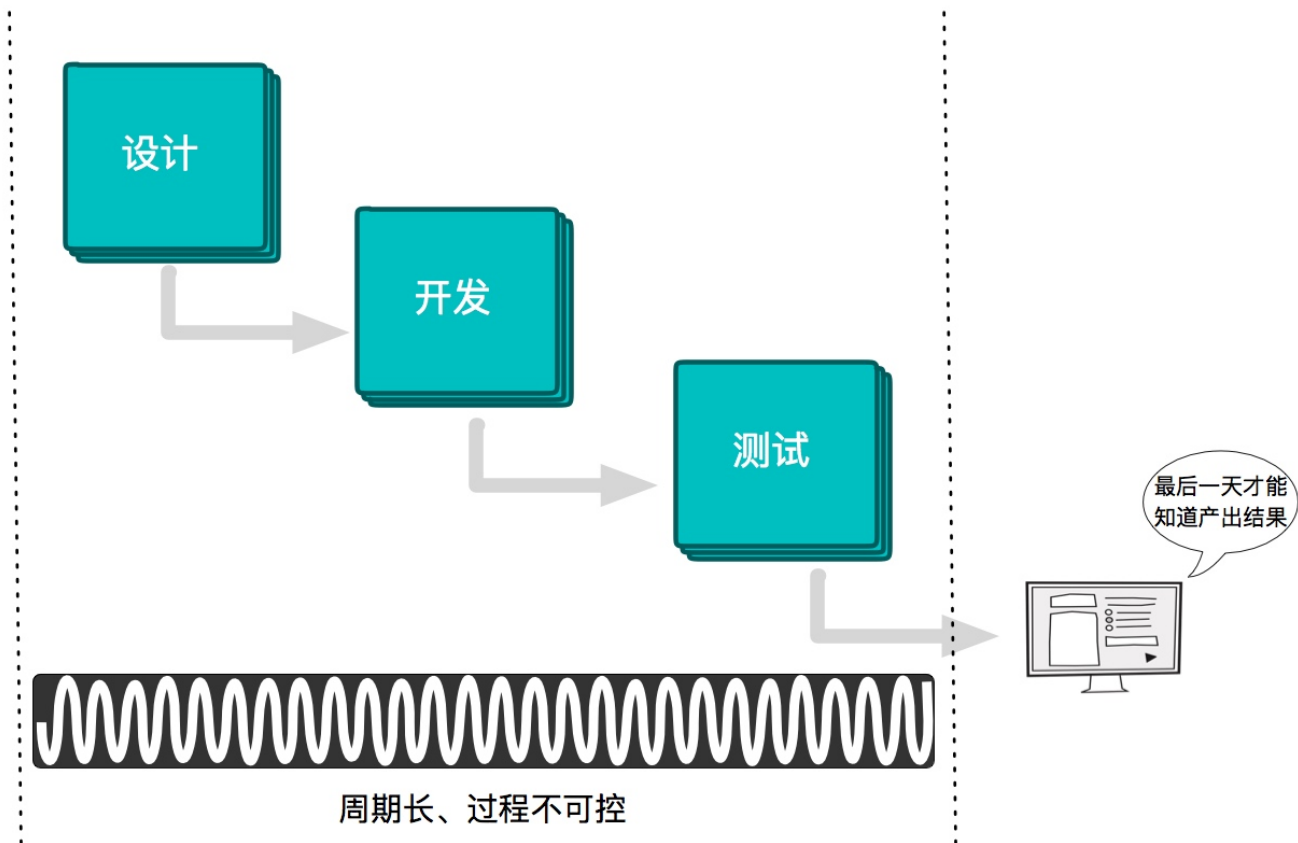
如果你的团队只有很有限的开发能力，那么怎么去做自动化，是做最原始的录制回放，还是数据驱动。复杂自动化也需要良好的基础设施支持。必须选择和团队，技能储备，基础设施与工具匹配的自动化策略。

持续集成与自动化测试相结合

持续集成的目的

传统瀑布存在的问题

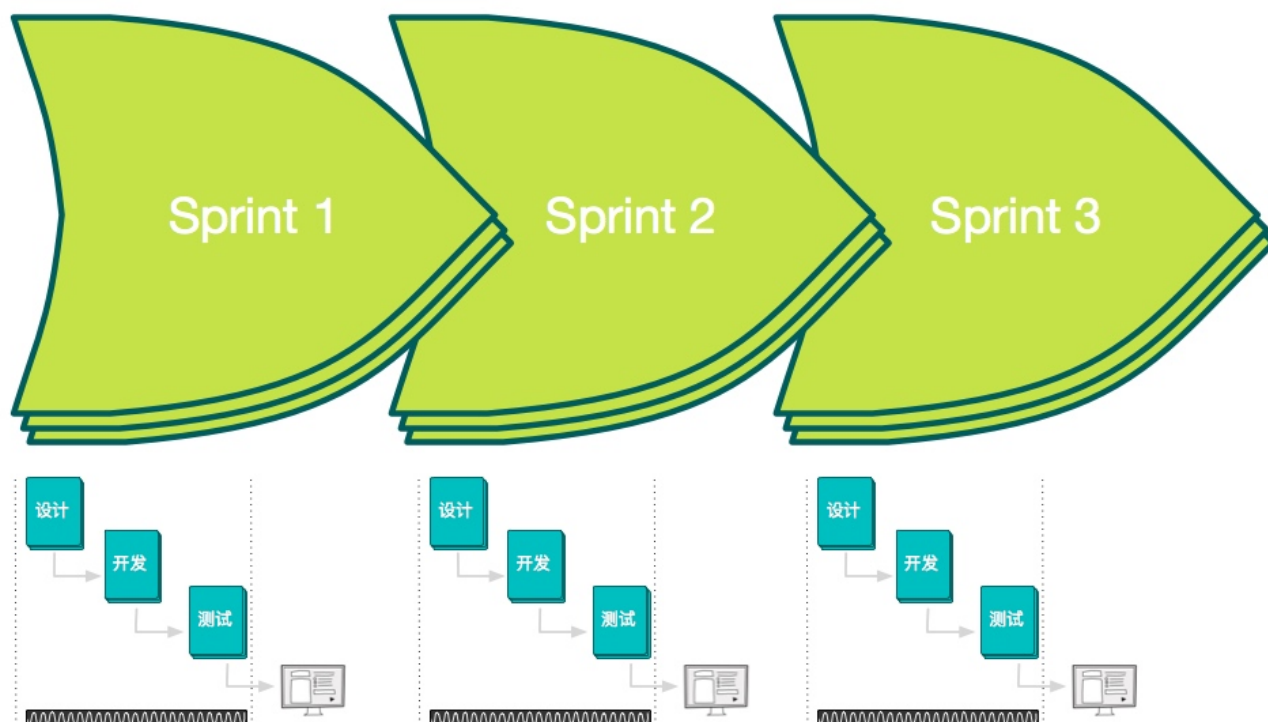
以移动应用开发为例子，因为移动应用的特点是开发周期、测试回归周期长，通常只有在最后一天才能知道这个app能不能通过验收，这样对公司而已风险很高。



船小好调头(小版本迭代)

为了避免这种风险，大部分公司对移动应用开发这块都会采用敏捷开发模式，不停做小版本迭代达到快速验收的目的。

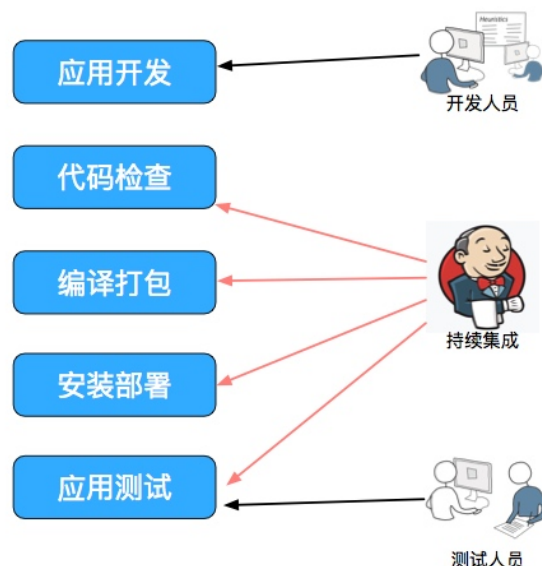
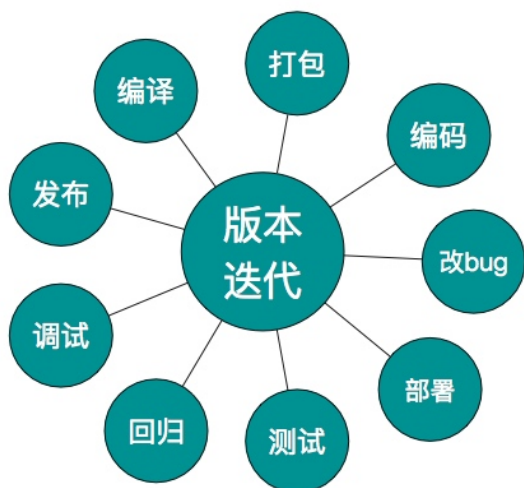
尽可能小版本迭代，缩短周期降低风险



持续集成与自动化测试的关系

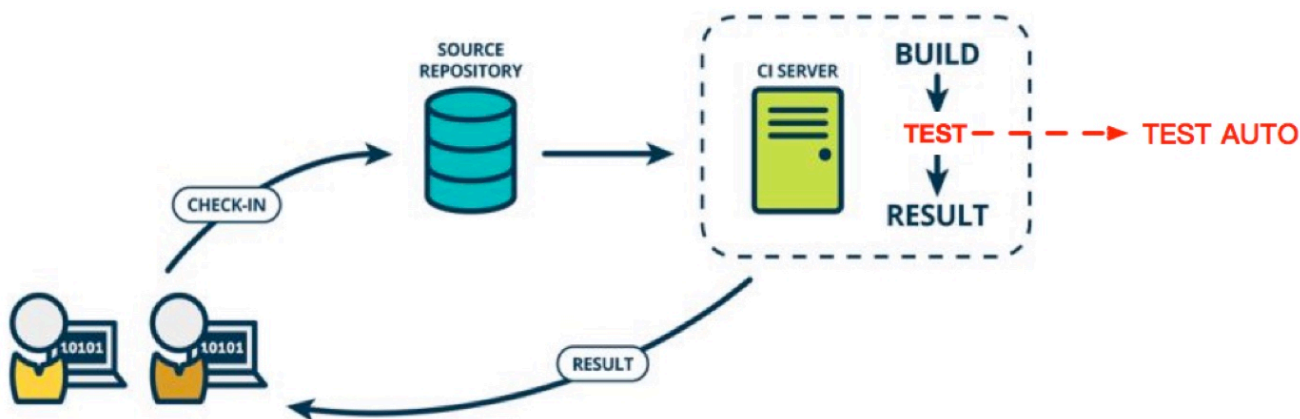
持续集成解决研发的噩梦

快速的小版本迭代降低了产品不可控的风险，但对于研发来说却产生了新的问题。假设你的老板要求每天至少迭代一个版本，这时候每天的工作状态可能就是“开发-编译-打包-测试”，结果就是重复且单一的工作占据了每天的工作。在这种情况下我们就需要引入持续集成与自动化测试，让系统每天自动的将成员提交的代码集成起来，释放更多的人力投入到更有价值的工作当中去。



持续集成为什么要引入自动化测试

前面我们提到了敏捷开发时需要每天自动的将成员提交的代码集成起来，通过系统自动的完成编译、构建、部署，但是没有办法及时的知道新代码和旧代码有没有集成成功。这时候就需要引入自动化测试，根据测试结果，我们可以确定新代码和原有代码能否正确地集成在一起。



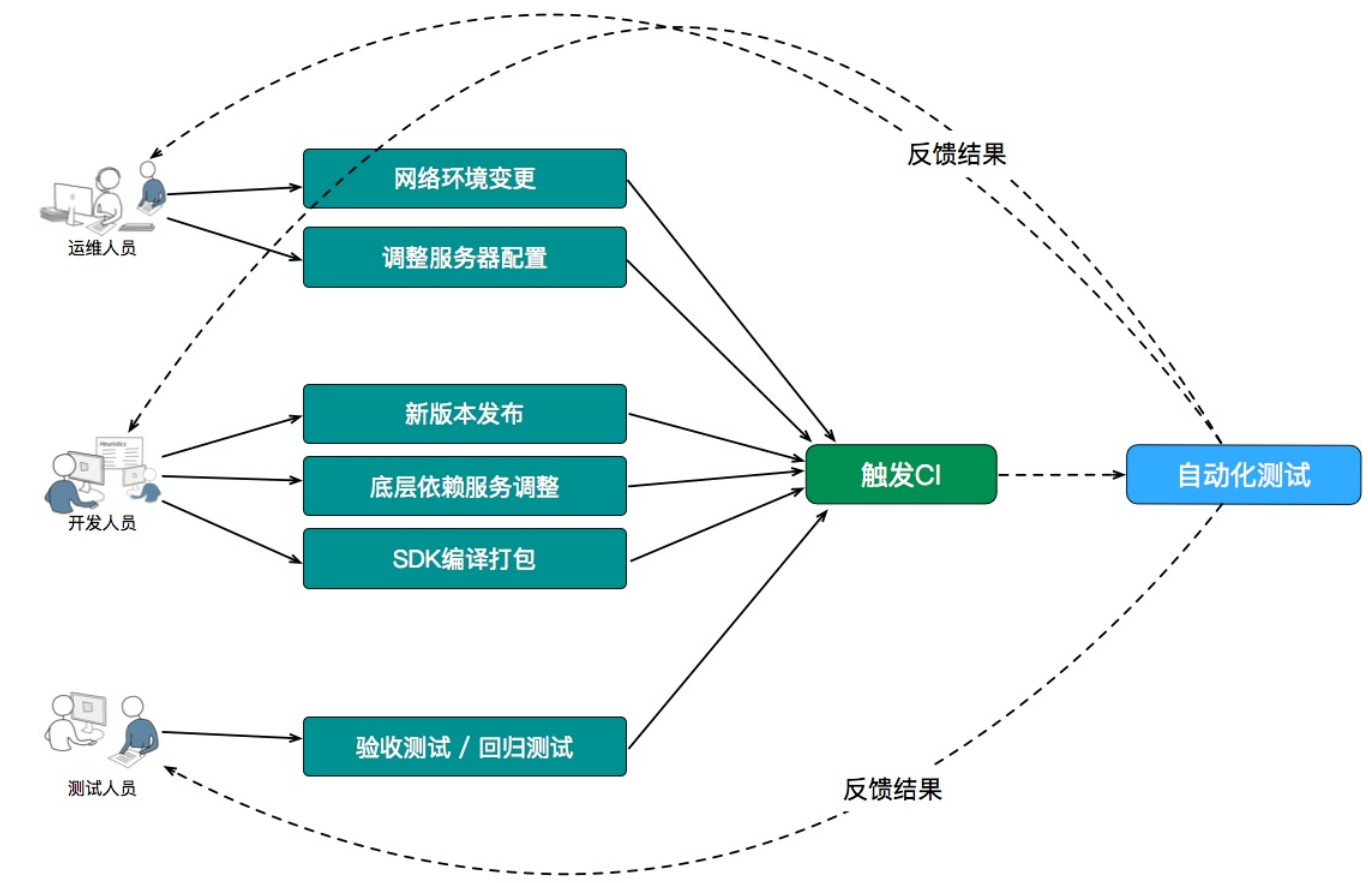
Jenkins做持续集成

Jenkins使用场景

Jenkins是一个广泛用于持续构建的可视化web工具，可以整合任何你想整合的内容。在本次的分享中，我们可以把刚刚完成的「《王国纪元》游戏自动化登陆测试」加入到

jenkins的构建任务中，集成到各个场景中。

比如运维人员普遍不熟悉业务，但却会经常调整服务器配置，所以每次调整无法直观的判断业务情况。当引入了jenkins后情况就不同了，运维在调整服务器后能够主动的从jenkins上获取到最新的业务检测报告。



将自动化测试集成到jenkins任务中

创建一个jenkins任务，用于执行上面写好的python脚本。之后可以在需要时手动或自动的运行自动化测试，并将测试结果以报告的形式反馈结果。

- 返回到工程
- 状态集
- 变更记录
- 控制台输出
- 文本方式查看
- 编辑编译信息
- 删除本次生成
- 参数
- Open Blue Ocean
- 前一次构建

控制台输出

```
Started by user root
Building in workspace /Users/Shared/Jenkins/Home/workspace/ln-game-test
Set build name.
New build name is 'LN-TEST-48'
[ln-game-test] $ /bin/sh -xe /Users/Shared/Jenkins/tmp/jenkins2493778275626004098.sh
+ python test.py

+-----+
+ CASE: 启动游戏
+-----+
+ TIME: 12.68s
+ RETURN: success
+-----+

+ CASE: 判断是否进入游戏主界面
+-----+
+ find: is_main.png 0.371137946844 (239, 216)
+ Can't find is_main.png (max:0.371137946844 expect:0.9)
+ find: btn_close_full.png 0.949985861778 (1209, 28)
+ find: btn_close_full.png 0.355695396662 (399, 628)
+ Can't find btn_close_full.png (max:0.355695396662 expect:0.8)
+ find: is_main.png 0.846453667082 (5, 0)
+ TIME: 7.2s
+ RETURN: success
+-----+
```

Build History	构建历史
find	X
LM-TEST-48 2018-4-2 下午4:34 结果: success 耗时: 68.86s 报告: log/20180402163415/index.html	
LM-TEST-47 2018-4-2 下午4:23 结果: success 耗时: 24.27s 报告: log/20180402162340/index.html	
LM-TEST-46 2018-4-2 下午4:23 结果: error 耗时: 24.36s 报告: log/20180402162316/index.html	


将测试的结果以报告形式反馈用户

Product Test Report 《Lords Mobile》


Time:	2018-03-30 13:59:13
result:	successful
Device:	AndyOSX
System:	Android
Version:	4.2.2
Cpu Name:	Intel(R) Core(TM) i7-4770HQ CPU @ 2.20GHz
Cpu Process:	4
Memory:	1273752k

启动游戏

判断是否进入游戏主界面

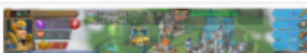


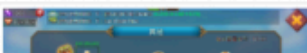
0.37ls_main.png

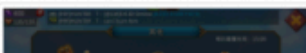


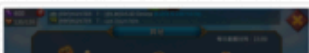
0.85ls_main.png

faceBook登陆游戏







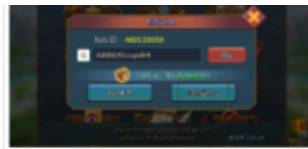




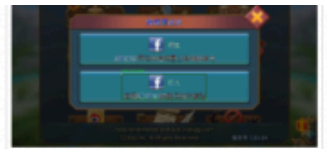
0.88 btn_setting.png



0.93 btn_setting_account.png



0.87 btn_setting_account_select01.png



0.87 btn_setting_account_select02.png



0.91 btn_setting_account_select03.png



0.75 btn_setting_account_select04.png



0.96 btn_setting_account_wait01.png



1.0 account_fb_login02.png



0.88 login_success.png

启动游戏

判断是否进入游戏主界面



0.39 is_main.png



0.89 is_main.png

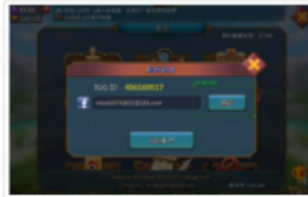
注销登陆



0.85 btn_setting.png



0.93 btn_setting_account.png



0.95 btn_account_logout.png



0.89 logout_success.png