目 录 (上 篇)

优秀的测试用例应该有延展性……01

测试女巫之接口测试篇……03

互联网产品测试之挑战、工具和测试方法……32

软件测试者需要了解关于自动化的什么……39

安卓 APP 自动化测试之搞定界面元素……43

不可不知的数据分层测试……49

让我们成为开源软件测试者……52

小白们需知的测试流程……57

运用场景覆盖管理提升测试精细化管理的实践与展望……61

测试工作，你必须了解的那些事……66

自动化测试之 QTP 学习笔记……68

优秀的测试用例应该有延展性

◆ 作者：流氓贵族

听了公司大神的一个培训，讲的是 Shell 脚本编程，其实所有的编程语言的思路都是差不多的，语法上可能会有一些小的差别。对于不同的编程人员来讲，差异就在与编程人员自身的编程习惯和思考的是否全面，这些决定了程序的可读性和可重用性。大神说了一句话：“一个好的程序不在于功能有多炫，性能有多好，好的程序应该有很好的可读性和可重用性。其实学习知识并不难，难的是对知识的传承。”

我们写的测试用例也一样，可读性和可重用性也非常重要。测试的功能多了会发现所有的功能几乎都可以找到原型或者由几个原型组合。那么我们在写测试用例的时候就应该考虑到用例可读和重用两点，对于新的功能就不必每次都要重头设计测试用例，可以在原有用例的基础上修改和新增。复用测试用例可以在很大的程度上减少重复性的工作。

对于以上两种特性，我概括总结为延展性，即一个好的测试用例应该支持类似功能的复用，可以作为优化、延展功能的测试用例基础版本，这个体现在测试用例的可复用性。而要保证上述两项的要求，最基础的要保证测试用例可以被任何的测试人员读懂且无歧义，这个体现在测试用力的可读性。

那么如何保证测试用例的延展性？在可读性方面，部门内部可以执行一套测试用例的书写标准，有了统一的标准就可以很大程度上的避免由于测试工程师的个人书写习惯和风格导致的测试用例可读性低，例如有的测试工程师在写测试用例的时候喜欢用自己惯用的缩略词，导致其他测试工程师在看的时候不理解测试用例写的是什么。

在可复用性方面，建议在写测试用例的时候，着重突出功能的实现而非针对某一个特定的系统需求编写测试用例，这种编写的方法能够很大程度上的保证类似功能的测试用例复用；测试用例的编写思路建议按照业务流程，分不同场景来编写，这种编写方法能够方便在做优化、延展功能需求的时不必重新编写测试用例，将原有测试用例稍作更

改增加即可。

测试用例之于测试工程师就像代码之于开发工程师，要好好的维护和整理。严谨是测试的生命，分享是最好的学习方法，立即开始实践。附上一个我理解的测试用例的标准:

1. 完整性
2. 准确性
3. 描述清晰无歧义
4. 不冗余
5. 可读性
6. 可复用

测试女巫之接口测试篇

◆作者：王平平

摘要： 此模块是使用 Python 语言，讲解接口测试类型的项目如何根据我们已经学习模块进行分析，以及如何实现两种类型的 API 测试项目的自动化，这次主要讲解的是分析方法，以及如何根据实际项目的状况组织相应的算法。

一、前言：

接连几期我们讨论都是 Python 实用的第三方模块，这一次我们根据实际遇到的 API 类型的测试项目，讲解如何根据我们已经学习的知识，实现实际项目的自动化。这一次主要的侧重点主要是如何分析，如何将我们已经学习的知识去解决实际的问题。我会将这些解决问题的方法总结出来，最近听到一个非常有意思的说法：人类是一个具有超强抽象能力的物种，所以为了对得起“人类”这个称号，我也一直在不断的抽象，总结，希望找出方法，且这个方法希望是可以被推广可以解决读者遇到的实际问题。所以这一次的旅行是非常有意思的，就像金秋的收获季节一样，好了，让我们跟着测试女巫分析如何实现两种类型的 API 自动化：AT Command 以及 Json Command。

二、接口测试介绍

1、接口测试基本概念

是测试系统组件间接口的一种测试。主要用于检测外部系统与系统之间以及系统内部各个子系统之间的交互点。

2、接口测试的分类：

1) 系统与系统之间的调用

例如淘宝或者苏宁易购可以使用支付宝支付，则支付宝需要提供接口供淘宝或者苏宁易购使用。

2) 同一个系统中上层服务对下层服务的调用

例如一个路由器产品的底层通讯与操作界面之间的调用。

3、接口测试流程:

模拟客户端连接服务器(服务器提供的端口是否可访问)

客户端发送报文请求

服务器端接收请求并做处理

↓

检查返回的预期结果并与实际结果对比

↓

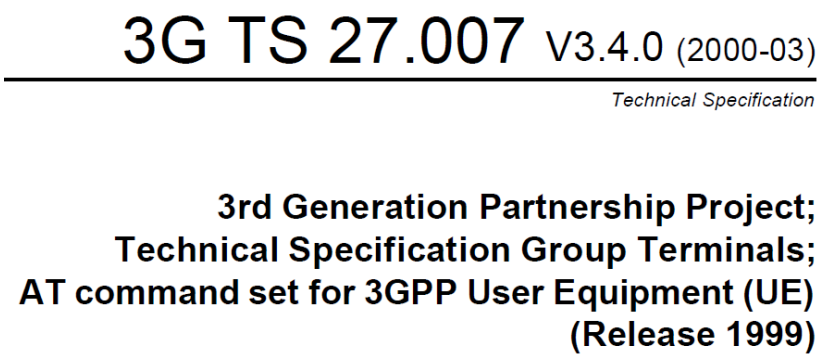
结束

4、AT Command

1) 基本资料

AT 的意义就是 Attention Command，它对于 3GPP 是有对应的技术标准文档如下图:

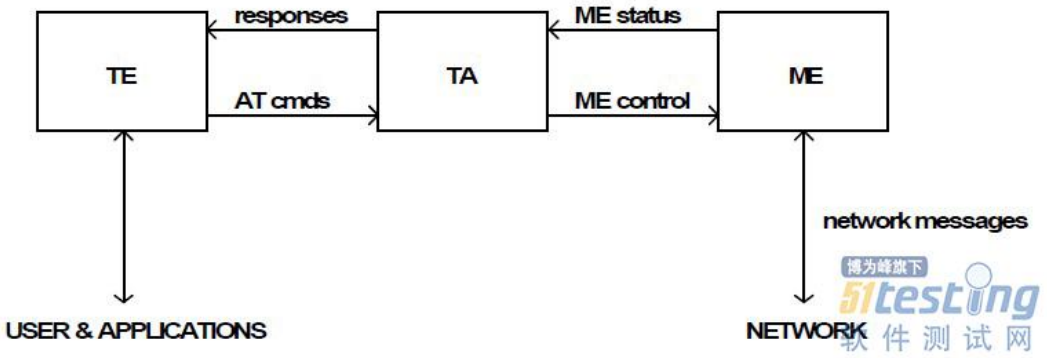
所以我们测试的 AT Command 往往是基本 3GPP 此份文档的基础上，进行的二次开发。


3G TS 27.007 V3.4.0 (2000-03)

Technical Specification

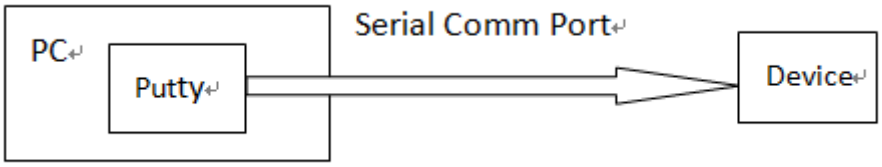
3rd Generation Partnership Project;
Technical Specification Group Terminals;
AT command set for 3GPP User Equipment (UE)
(Release 1999)

2) 3GPP 文档中提到的解释 AT Command 框架的图示:



用户可以通过 AT 命令进行待测物配置，状态控制，串口接口控制，安全控制，因特网服务控制，SIM card 相关控制，相关数据域名控制，标识控制等各方面的控制。

3) 测试框架



5、Json Command

1) 基本资料

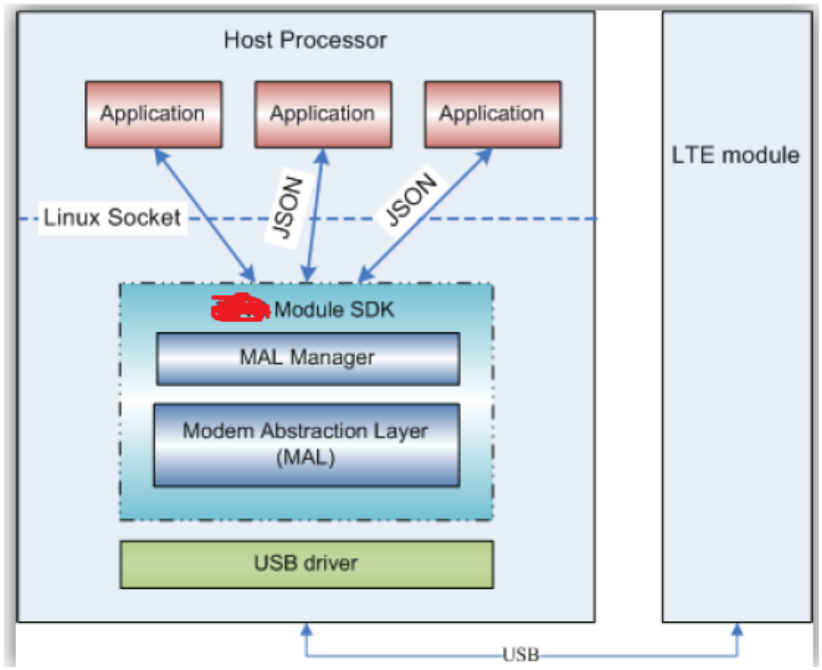
主要测试 Module 的 SDK 功能，SDK 的意义是 Software Development Kit（软件开发工具包）

2) 原理图以及说明

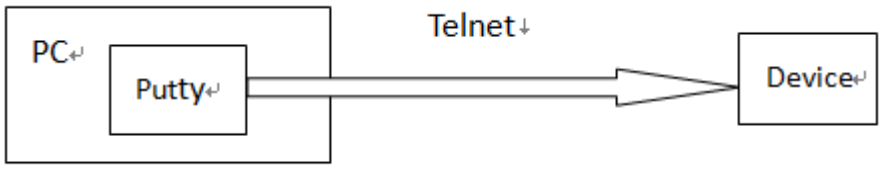
MAL 的意思是一组可以访问 Modem services 的一组应用核心库:

例如可以访问 SIM 卡的一些功能，网络的相关参数设置，无线数据功能，以及短消息功能等

MAL Manager 提供了一些管理服务器来控制模组的一些行为。例如；连接管理，短消息管理，模组监控管理，模组更新版本管理，配置管理等。



3) 测试框架

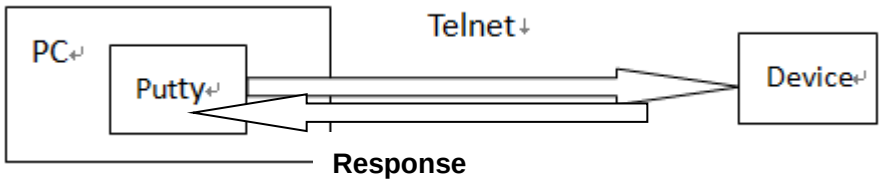


三、Json Command API

1、背景知识

Pywinauto 是适用 Window UI 自动化的模块，其实从它的名字就可以看出它的作用，“Py”代表 Python；“Win”代表可以控制 Window UI 上的软件；“Auto”代表可以自动化。

（一）【待测物分析】测试框架

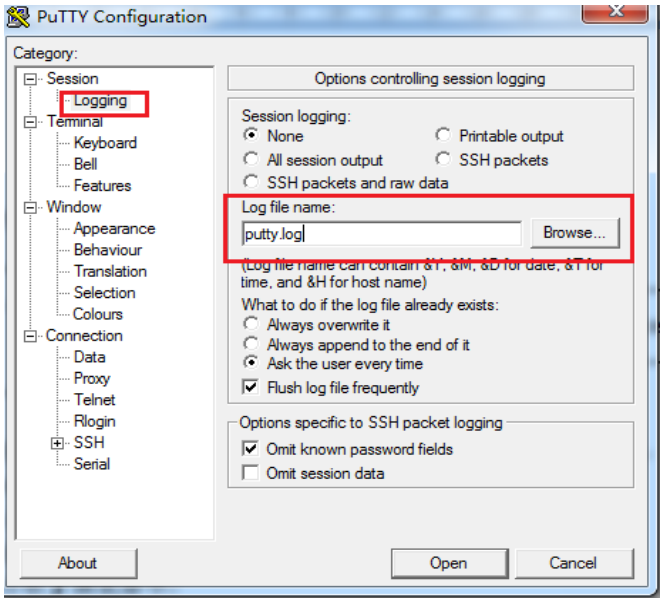


1）通过安装在 PC 上的 Putty 通过 Telnet 与 Device 建立连接

2）通过 Putty 向 Device 下命令（Input）

3）Device 给出 Response 并反馈给 Putty （Output）

4）Putty 会自动将 Log 保存到一个设定好的路径，如果保存默认路径则反馈的信息将会保存在与 Putty 此工具在同一个文件夹中。



（二）【待测物分析】测试命令

以一个命令为例进行说明:

Input 就是 JsonClient /tmp/cgi-2-sys get_wwan_serving_system_provider 注意 JsonClient /tmp/cgi-2-sys 是每个命令都要加的“头文件”而 get_wwan_serving_system_provider 是根据不同的命令会有不同的字串

Response 就是当前使用运营商的名称

2.1.2. get_wwan_serving_system_provider

This API is used to query the service provider currently in use.

■ Request arguments：

Name	Type	Requirement	Description

■ Response other values：

Name	Type	Requirement	Description
provider	String	Mandatory	Network provider name • Unknown - if network is not registered.

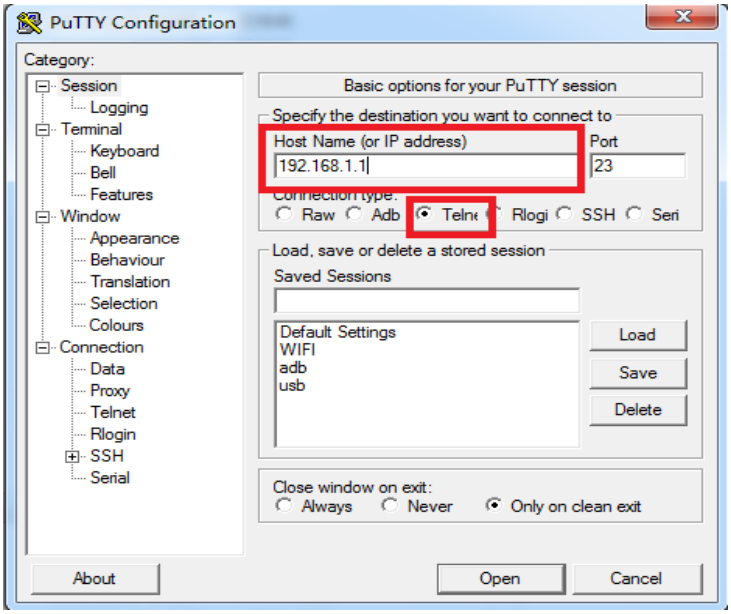
■ Example：

{ "action" : "get_wwan_serving_system_provider" }

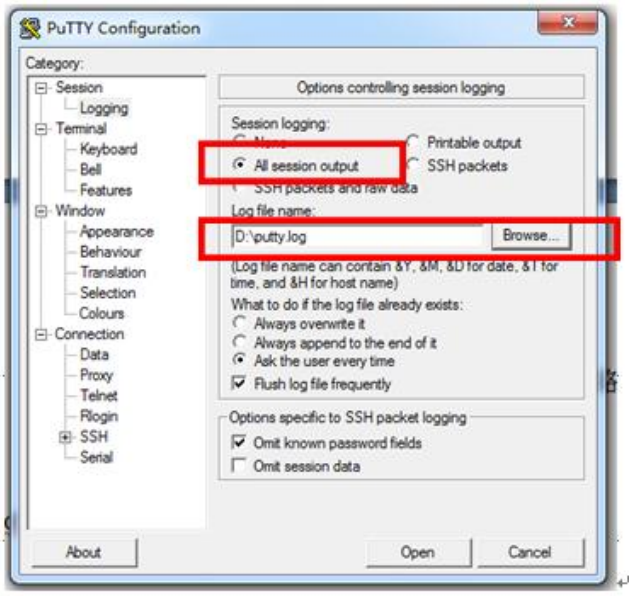
{ "get_wwan_serving_system_provider": { "errno": 0, "errmsg": "", "provider": "Chunghwa" } }

（三）【待测物分析】具体操作步骤分析

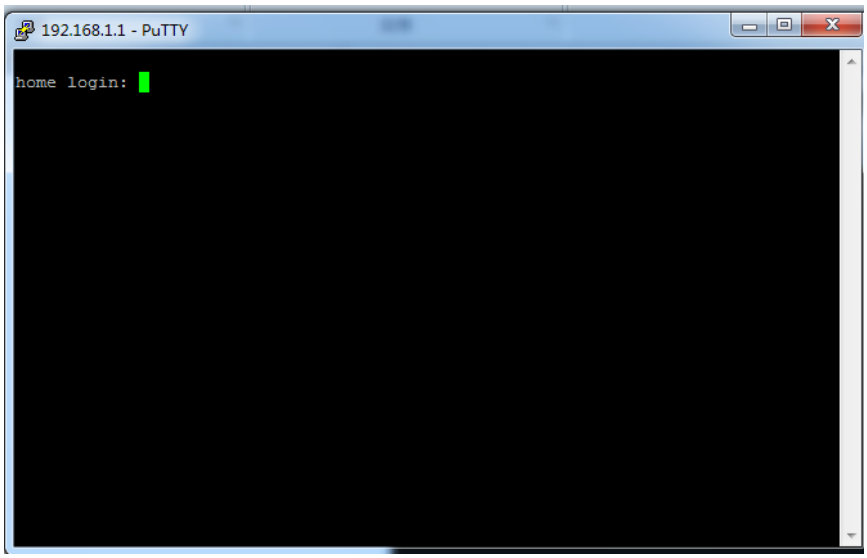
1）首先第一步到 Session 中，选择连接方式为 Telnet,且设置正确的 Host Name，其它为默认值



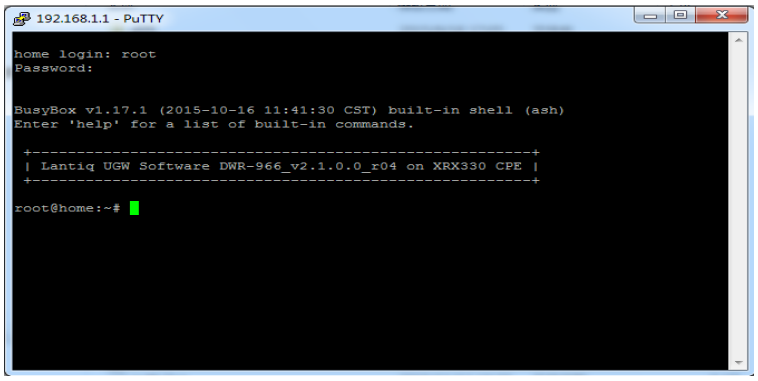
2）第二步到 Logging 中，选择 Log 存储的路径，以及选择 Log 保存的方式



3）点击 Open 后，出现如【图 3】的编辑界面，使用 TypeKey 此函数输入用户名以及密码



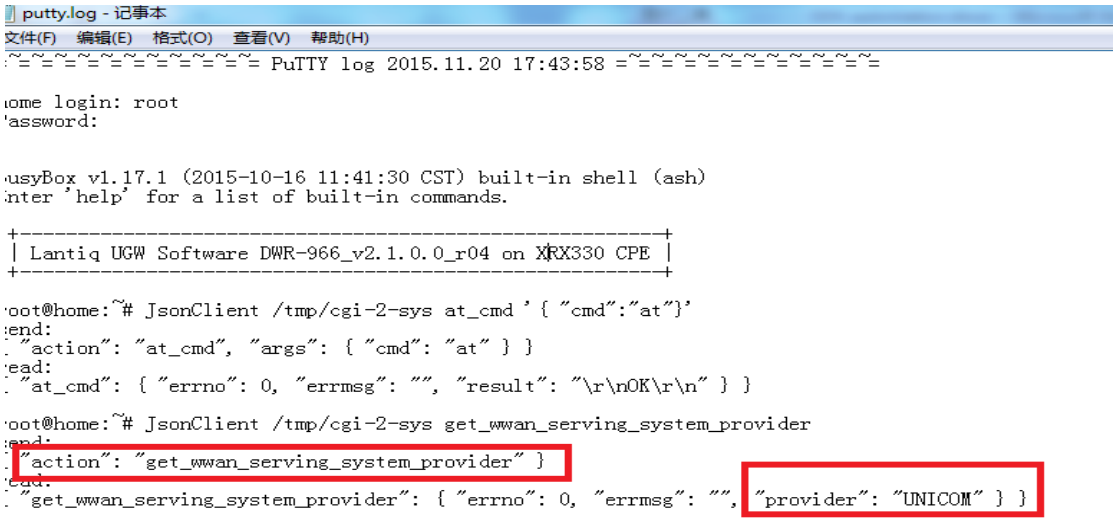
4）出现如下界面，使用 k.type_string 输入命令



5）测试完毕后，需要到 D 盘根目录下寻找 Putty.log 此文件

看到此文档就可以看出 Putty 工具的好处，它是将所有 Input 以及 Output 全部都忠实地记录下来，如下图:

接下来就要在此 putty.log 中提炼出 response，并将此 response 与已知的数值进行对比。



2、需要的 Python 的知识分析

1）Pywinauto 解决控制 Putty tool 上各种控件的问题：Python 系列课程已解决。

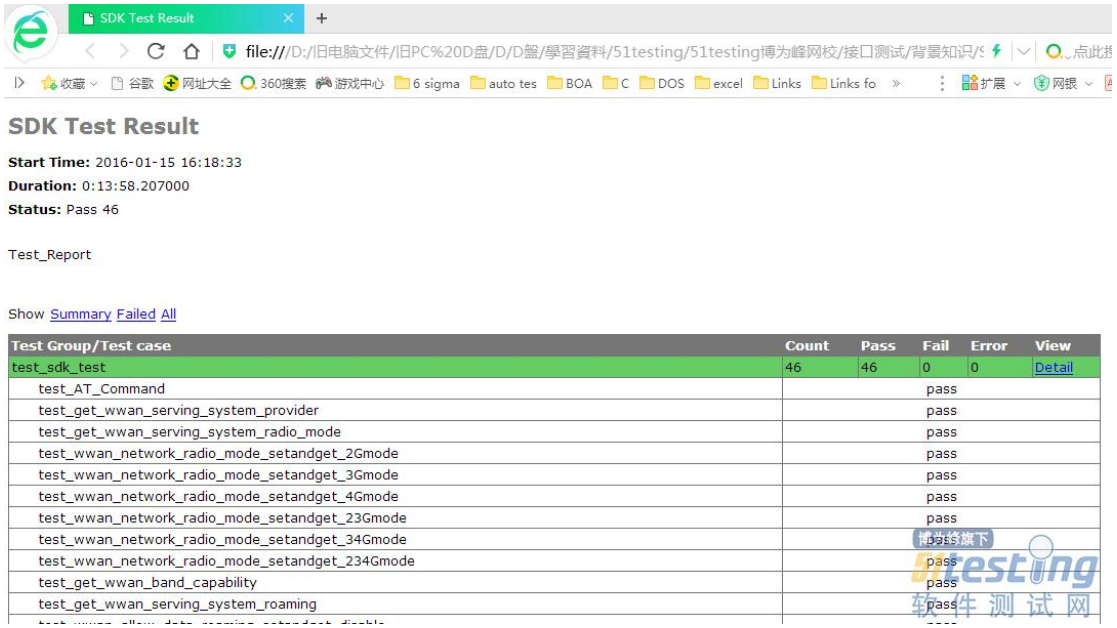
2）Pykeyboard 解决在类似 dos 界面输入命令的问题：Python 系列课程已解决。

3）Unittest 解决搭建测试框架，因为测试命令很多，不可能一个 command 一个脚本，如下：Python 系列课程已解决。

2.1. WWAN 7
2.1.1. get_wwan_serving_system_status 7
2.1.2. get_wwan_serving_system_provider 8
2.1.3. get_wwan_serving_system_radio_mode 9
2.1.4. get_wwan_network_radio_mode 9
2.1.5. set_wwan_network_radio_mode 10
2.1.6. get_wwan_radio_info 10
2.1.7. get_wwan_band_capability 11
2.1.8. get_wwan_band_info 12
2.1.9. get_wwan_lte_band_info 12
2.1.10. get_wwan_lte_ca_info 13
2.1.11. get_wwan_serving_system_roaming 14
2.1.12. get_wwan_allow_data_roaming 14
2.1.13. set_wwan_allow_data_roaming 14
2.1.14. get_wwan_network_time 15
2.1.15. get_wwan_ipv4_network_state 15
2.1.16. get_wwan_ipv4_network_ip 16
2.1.17. get_wwan_ipv4_network_connection_time 16
2.1.18. get_wwan_ipv4_network_status 17
2.1.19. get_wwan_ipv6_network_state 18

4）对于 putty.log 的操作：读取指定的位置的字串，并将字串提取出来：此问题需要学习 Python 对于字串的处理。

5）将上一步提取的字串与已知的字串对比根据最后的对比的结果产出一个比较漂亮的报告：Python 系列课程已解决。



3、Python 对于字串处理的详细分析

我们先分析待处理的 Log 再决定我们需要学习什么样的函数（学习一定要有针对性）

1) 首先我们需要将 putty.log 此文档使用代码将其打开

先找寻 Python 官网，建议大家可以到网上搜“超级无敌 Python 教程”此教程几乎是将官网翻译一遍，如果真的不想看这么多的英文，可以看这个教材。

在此文档中找到了相关的函数：open (filename, mode)

经过实践 file name 是文本文档所在的路径+file name

```
f=open('d:\\SDK/putty_WWAN&Network&System.log','r')
```

open() 返回一个文件对象 ，通常的用法需要两个参数： “open(*filename*, *mode*)”。

```
>>> f=open('/tmp/workfile', 'w')
>>> print f
<open file '/tmp/workfile', mode 'w' at 80a0960>
```

第一个参数是一个标识文件名的字符串。第二个参数是由有限的字母组成的字符串，描述了文件将会被如何使用。可选的模式有： ‘r’，此选项使文件只读； ‘w’ ，此选项使文件只写（对于同名文件，该操作使原有文件被覆盖）； ‘a’ ，此选项以追加方式打开文件； ‘r+’ ，此选项以读写方式打开文件；如果没有指定，默认为 ‘r’模式。

在Windows 和 Macintosh平台上， ‘b’模式以二进制方式打开文件，所以可能会有类似于 ‘rb’ ， ‘wb’ ， ‘r+b’等等模式组合。 Windows平台上文本文件与二进制文件是有区别的，读写文本文件时，行尾会自动添加行结束符。 这种后台操作方式对文本文件没有什么问题，但是操作JPEG或EXE这样的二进制文件时就会产生破坏。在操作这些文件时一定要记得以二进制模式打开。（需要注意的是Mactiontosh平台上的文本模式依赖于其使用的底层C库）。

2) 读取 putty.log 此文档中的信息

读取文本文件有三个方法，建议使用效率较高的 f. readlines()

一次将文件的所有文字读取出来，以列表的形式保存到一个变量中。

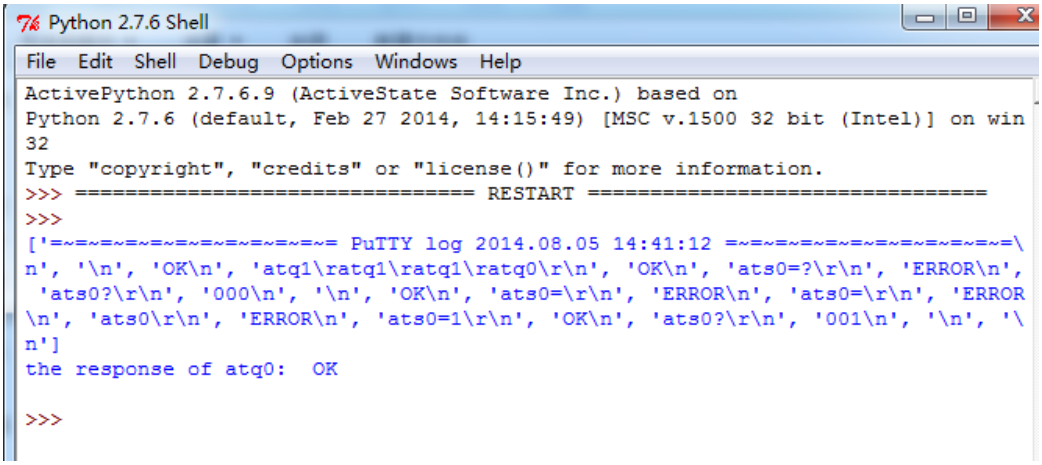
要读取文件内容，需要调用 f.read(*size*)，该方法读取若干数量的数据并以字符串形式返回其内容，字符串长度为数值*size*所指定的大小。如果没有指定size或者指定为负数，就会读取并返回整个文件。当文件大小为当前机器内存两倍时，就会产生问题。正常情况下，会按size尽可能大的读取和返回数据。如果到了文件末尾，`f.read()` 会返回一个空字符串（""）。

```
>>> f.read()
'This is the entire file.\n'
>>> f.read()
''
```

`f.readline()` 从文件中读取单独一行，字符串结尾会自动加上一个换行符，只有当文件最后一行没有以换行符结尾时，这一操作才会被忽略。这样返回值就不会有什么混淆不清，如果 if `f.readline()` 返回一个空字符串，那就表示到达了文件末尾，如果是一个空行，就会描述为 '\n'，一个只包含换行符的字符串。

```
>>> f.readline()
'This is the first line of the file.\n'
>>> f.readline()
'Second line of the file\n'
>>> f.readline()
''
```

`f.readlines()` 返回一个列表，其中包含了文件中所有的数据行。如果给定了 *sizehint* 参数，就会读入多于一行的比特数，从中返回行列表。这个功能通常用于高效读取大型行文件，避免了将整个文件读入内存。这种操作只返回完整的行。



3) 从将列表形式的字串转成 String 类型的字串

‘,’.join (content) 即对于列表的元素以逗号分隔的方式连接在一起，并将其变为字符串。

创建列表：

```
>>> music = ["Abba","Rolling Stones","Black Sabbath","Metallica"]
>>> print music
```

输出：

```
['Abba', 'Rolling Stones', 'Black Sabbath', 'Metallica']
```

通过join函数通过空格连接列表中的元素：

```
>>> print ' '.join(music)
```

返回结果

```
Abba Rolling Stones Black Sabbath Metallica
```

例子： 第一块看不懂的内容就是直接将 readlines()读取到的内容打印出来的结果，第二块是通过 Join 函数转换过的内容。

```
import sys
import os

sys.path.append(os.path.dirname(__file__))

fsock=open('e:\\1.log','r')
content=fsock.readlines()
print content
contentstring=','.join(content)
print contentstring
```

```
['\n', '\xd5\xfd\xd4\xda Ping 192.168.1.1 \xbe\xdf\xd3\xd0 32 \xd7\xd6\xbd\xda\x
b5\xc4\xca\xfd\xbe\xdd:\n', '\xc0\xb4\xd7\xd4 192.168.1.100 \xb5\xc4\xbb\xd8\xb8
\xb4: \xce\xde\xb7\xa8\xb7\xc3\xce\xca\xc4\xbf\xb1\xea\xd6\xf7\xbb\xfa\xa1\xa3\n
', '\xc0\xb4\xd7\xd4 192.168.1.100 \xb5\xc4\xbb\xd8\xb8\xb4: \xce\xde\xb7\xa8\xb
7\xc3\xce\xca\xc4\xbf\xb1\xea\xd6\xf7\xbb\xfa\xa1\xa3\n', '\xc0\xb4\xd7\xd4 192.
168.1.100 \xb5\xc4\xbb\xd8\xb8\xb4: \xce\xde\xb7\xa8\xb7\xc3\xce\xca\xc4\xbf\xb1
\xea\xd6\xf7\xbb\xfa\xa1\xa3\n', '\xc0\xb4\xd7\xd4 192.168.1.100 \xb5\xc4\xbb\xd
8\xb8\xb4: \xce\xde\xb7\xa8\xb7\xc3\xce\xca\xc4\xbf\xb1\xea\xd6\xf7\xbb\xfa\xa1\
xa3\n', '\n', '192.168.1.1 \xb5\xc4 Ping \xcd\xb3\xbc\xc6\xd0\xc5\xcf\xa2:\n', '
    \xca\xfd\xbe\xdd\xb0\xfc: \xd2\xd1\xb7\xa2\xcb\xcd = 4\xa3\xac\xd2\xd1\xbd\x
d3\xca\xd5 = 4\xa3\xac\xb6\xaa\xca\xa7 = 0 (0% \xb6\xaa\xca\xa7)\xa3\xac\n', '\n
']

,正在 Ping 192.168.1.1 具有 32 字节的数据:
,来自 192.168.1.100 的回复: 无法访问目标主机。
,来自 192.168.1.100 的回复: 无法访问目标主机。
,来自 192.168.1.100 的回复: 无法访问目标主机。
,来自 192.168.1.100 的回复: 无法访问目标主机。
,
,192.168.1.1 的 Ping 统计信息:
,    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
,
```

4) 从第三步读取的信息提取希望想要的资料

A、观察命令的反馈结果: 每个 Response 都有一个规律: 都有“errmsg”这个字串，这个字串后面的字串就是命令的反馈结果，我们就需要根据这个反馈结果来判断是否为 Pass 还是 Fail，如下图:

```
root@home:~# JsonClient /tmp/cgi-2-sys at_cmd '{ "cmd":"at"}'
send:
{ "action": "at_cmd", "args": { "cmd": "at" } }
read:
{ "at_cmd": { "errno": 0, "errmsg": "", "result": "\r\nOK\r\n" } }

root@home:~# JsonClient /tmp/cgi-2-sys get_wwan_serving_system_provider
send:
{ "action": "get_wwan_serving_system_provider" }
read:
{ "get_wwan_serving_system_provider": { "errno": 0, "errmsg": "", "provider": "UNICOM" } }

root@home:~# JsonClient /tmp/cgi-2-sys get_wwan_serving_system_radio_mode
send:
{ "action": "get_wwan_serving_system_radio_mode" }
read:
{ "get_wwan_serving_system_radio_mode": { "errno": 0, "errmsg": "", "radio_mode": 4 } }

root@home:~# JsonClient /tmp/cgi-2-sys set_wwan_network_radio_mode '{ "radio_mod
e":2}'
send:
{ "action": "set_wwan_network_radio_mode", "args": { "radio_mode": 2 } }
read:
{ "set_wwan_network_radio_mode": { "errno": 0, "errmsg": "" } }

root@home:~# JsonClient /tmp/cgi-2-sys get_wwan_network_radio_mode
send:
{ "action": "get_wwan_network_radio_mode" }
read:
{ "get_wwan_network_radio_mode": { "errno": 0, "errmsg": "", "radio_mode": 2 } }

root@home:~# JsonClient /tmp/cgi-2-sys set_wwan_network_radio_mode '{ "radio_mod
e":3}'
```

B、如何找到 errmsg 后面的字串？(即先定位)

pattern=re.compile(r'errmsg.*')

即将正则表达式编译为一个正则表达式对象。

*称之为“贪婪模式”，例如 ca*t 将匹配 "ct" (0 个 "a" 字符), "cat" (1 个 "a"), "caaat" (3 个 "a" 字符)等等，对于我们建立的这个正则表达式的意义是：抓取 errmsg 后匹配任意除了换行符以外的字符。

所以建立的这个 Patten 的意义就是尽量多得将 errmsg 后面的字符匹配出来。

C、定位完成后开始查找

```
location_index=contentstring.find('errmsg',start)
```

Start 是从第一个位置开始寻找（对于已经测试过的命令也要重新开始寻找）

描述

Python find() 方法检测字符串中是否包含子字符串 str ，如果指定 beg（开始） 和 end（结束） 范围，则检查是否包含在指定范围内，如果包含子字符串返回开始的索引值，否则返回-1。

语法

find()方法语法：

```
str.find(str, beg=0, end=len(string))
```

参数

- str -- 指定检索的字符串
- beg -- 开始索引，默认为0。
- end -- 结束索引，默认为字符串的长度。

返回值

如果包含子字符串返回开始的索引值，否则返回-1。

D、实现基本功能：定位和查找函数找到了接下来需要考虑一下逻辑，即很多命令执行

下来有很多 errmsg 该如何处理？

寻找字串中所有 errmsg 所在的位子，并将其放到 location_index 中

通过 While 循环语句将 errmsg 的 index 一一放到 location 数组中

```
location=[]
    while True:
        location_index=contentstring.find('errmsg',start)
        if location_index==-1:
            break
        start=location_index+1
        location.append(location_index)
```

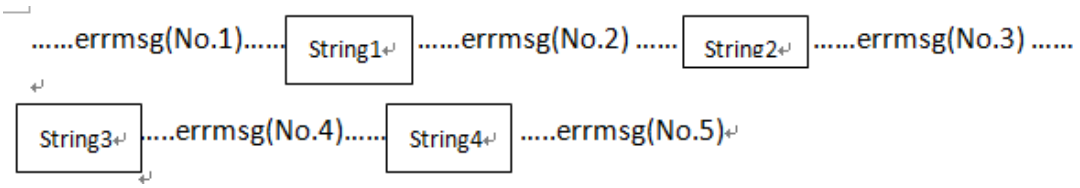
E、因为建立的 Pattern 是返回 errmsg 后面所有的字串

而 “findall()” 就是找到 RE 匹配的所有子串，并把它们作为一个列表返回。

然后判断正确的结果是否在 findall 截取的字串中，如果在的话，就是 Pass 否则就是 Fail。

```
#get_wwan_serving_system_provider(insert 4G SIM card)
def get_wwan_serving_system_provider(self,k):
    sys.path.append(os.path.dirname(__file__))
    k.type_string('JsonClient /tmp/cgi-2-sys get_wwan_serving_system_provider')
    time.sleep(2)
    k.tap_key(k.enter_key)
    time.sleep(2)
    f=open(filepath,'r')
    content=f.readlines()
    contentstring=','.join(content)
    pattern=re.compile(r'errmsg.*')
    start=0
    location=[]
    while True:
        location_index=contentstring.find('errmsg',start)
        if location_index==-1:
            break
        start=location_index+1
        location.append(location_index)
    result=pattern.findall(contentstring,location[1])
    search_result=wwan_serving_system_provider_pass in result[0]
    self.assertTrue(search_result)

    time.sleep(2)
    f.close()
```

画简易的流程图说明如何抽象上述算法:



“errmsg”是需要找到的关键字, ”No.x”是 errmsg 所在的位置，stringx 是我们需要提炼的 response 字串。

Location=[No.1,No.2,No.3,No.4…..]

Result=位置为 No.x 的 errmsg 后面的字串

也就是说，我们需要知道所需要找到 response 是哪个位置，即 No.x 是固定的，然后我们需要使用 findall(string)，在读取的字串中，截取 No.x 后面的所有字串(也就是说把 No.x 前的字串清除掉)。

Question:

第一个 case 执行完毕后，使用 Unit test 框架执行下一个 case，如下，

这个架构有两个问题:

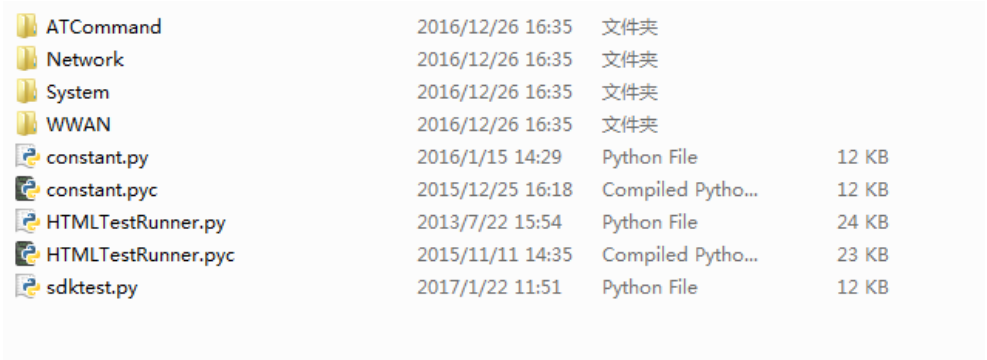
第一个问题：每个命令都要将 errmsg 的位置全部重新读一遍，且要放到数组中，后续抓取的 response 的相关字串只是数组中的一个位置中的字串而已。

第二个问题：findall 函数中，需要提供的第二个 parameter 的确切位置，这个位置不同的命令是会变化的，所以在安排测试命令时要注意顺序一定是固定的，例如 Command1 的位置是 location[1]，Command2 的位置就是 Location[2]

```
#get_wwan_serving_system_radio_mode(insert 4G SIM card)
def get_wwan_serving_system_radio_mode(self,k):
    sys.path.append(os.path.dirname(__file__))
    k.type_string('JsonClient /tmp/cgi-2-sys get_wwan_serving_system_radio_mode'
    time.sleep(2)
    k.tap_key(k.enter_key)
    time.sleep(3)
    f=open(filepath,'r')
    content=f.readlines()
    contentstring=','.join(content)
    pattern=re.compile(r'errmsg.*')
    start=0
    location=[]
    while True:
        location_index=contentstring.find('errmsg',start)
        if location_index==-1:
            break
        start=location_index+1
        location.append(location_index)
    result=pattern.findall(contentstring,location[2])
    search_result=wwan_serving_system_radio_mode_pass in result[0]
    self.assertTrue(search_result)

    time.sleep(2)
    f.close()
```

4、实际代码架构:

1）代码整体的组织:



2）Sdktest.py 代码说明

可以理解各个 function 测试的入口，即这里主要是搭建 Unittest 框架

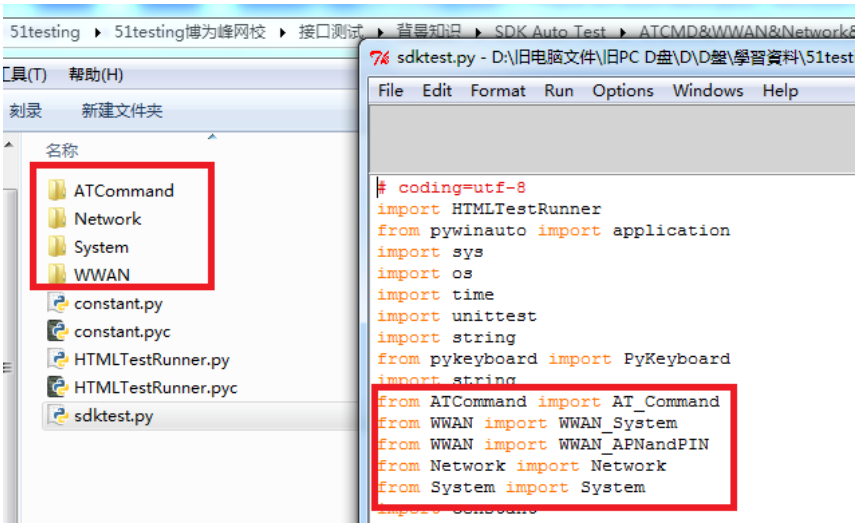
```
#执行测试的类
class test_sdk_test(unittest.TestCase):
    def setUp(self):
        pass

    ################################################################################
    #no1.AT Command
    def test_AT_Command(self):
        AT_Command.at_command(self,k)

    #no2.WWAN
    def test_get_wwan_serving_system_provider(self):
        WWAN_System.get_wwan_serving_system_provider(self,k)

    def test_get_wwan_serving_system_radio_mode(self):
        WWAN_System.get_wwan_serving_system_radio_mode(self,k)

    def test_wwan_network_radio_mode_setandget_2Gmode(self):
        WWAN_System.wwan_network_radio_mode_setandget_2Gmode(self,k)

    def test_wwan_network_radio_mode_setandget_3Gmode(self):
        WWAN_System.wwan_network_radio_mode_setandget_3Gmode(self,k)

    def test_wwan_network_radio_mode_setandget_4Gmode(self):
        WWAN_System.wwan_network_radio_mode_setandget_4Gmode(self,k)

    def test_wwan_network_radio_mode_setandget_23Gmode(self):
        WWAN_System.wwan_network_radio_mode_setandget_23Gmode(self,k)

    def test_wwan_network_radio_mode_setandget_34Gmode(self):
        WWAN_System.wwan_network_radio_mode_setandget_34Gmode(self,k)

    def test_wwan_network_radio_mode_setandget_234Gmode(self):
        WWAN_System.wwan_network_radio_mode_setandget_234Gmode(self,k)

    def test_get_wwan_band_capability(self):
```

3）constant.py 说明

对于一些经常需要变化的变量，在这里汇集，以函数的方式返回值，为了后续维护方便。

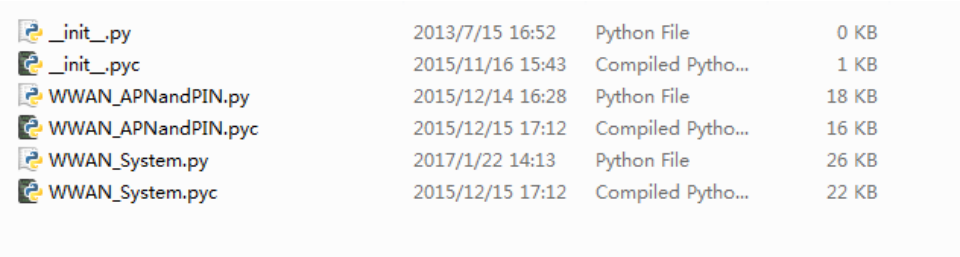
4）各个 Function 的脚本的调用逻辑

以一个大块功能一个文件夹的形式存在，而在 sdktest.py 此文件夹首先先把各个功能的脚本从这些文件夹中 import 进来，然后才能调用其中的函数



5）各个 function 脚本的设计逻辑

各个 function 文件夹中有几个脚本就需要 import 几次，例如 WWAN 文件夹中有两个脚本就要 import 两次。



每个脚本，对于每个命令建立一个函数，此函数后续会在 sdktest.py 此脚本中的 unittest 框架中以独立的 test case 被执行。

File Edit Format Run Options Windows Help

```
testcase=__name__.encode('utf-8')

#get_wwan_serving_system_provider(insert 4G SIM card)
def get_wwan_serving_system_provider(self,k):
    sys.path.append(os.path.dirname(__file__))
    k.type_string('JsonClient /tmp/cgi-2-sys get_wwan_serving_system_provider')
    time.sleep(2)
    k.tap_key(k.enter_key)
    time.sleep(2)
    f=open(filepath,'r')
    content=f.readlines()
    contentstring=','.join(content)
    pattern=re.compile(r'errmsg.*')
    start=0
    location=[]
    while True:
        location_index=contentstring.find('errmsg',start)
        if location_index==-1:
            break
        start=location_index+1
        location.append(location_index)
    result=pattern.findall(contentstring,location[1])
    search_result=wwan_serving_system_provider_pass in result[0]
    self.assertTrue(search_result)

    time.sleep(2)
    f.close()

#get_wwan_serving_system_radio_mode(insert 4G SIM card)
def get_wwan_serving_system_radio_mode(self,k):
    sys.path.append(os.path.dirname(__file__))
    k.type_string('JsonClient /tmp/cgi-2-sys get_wwan_serving_system_radio_mode')
    time.sleep(2)
    k.tap_key(k.enter_key)
    time.sleep(3)
    f=open(filepath,'r')
    content=f.readlines()
```

5、与实际 Test case 对比

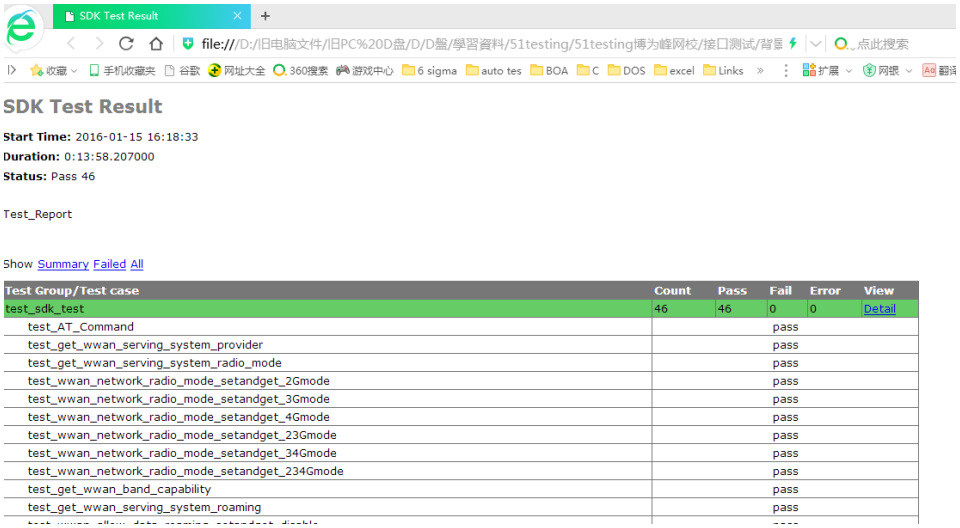
SDK Test Case

(一)Summary						
Project Name:						
SW Firmware:						
HW Firmware:						
Tester:						
Test Date:						
Test Result(Pass/Fail):						
Remark:						
(二)Test Item List						
Function		Test Result				Remark
		Pass	Fail	NA	NT	
1	WWAN					
2	Network					
3	System					
4	AT CMD					

```
time.sleep(2)
#执行测试的类
class test_sdk_test(unittest.TestCase):
    def setUp(self):
        pass

    ################################################################################
    #no1.AT Command
    def test_AT_Command(self):
        AT_Command.at_command(self,k)

    #no2.WWAN
    def test_get_wwan_serving_system_provider(self):
        WWAN_System.get_wwan_serving_system_provider(self,k)

    def test_get_wwan_serving_system_radio_mode(self):
        WWAN_System.get_wwan_serving_system_radio_mode(self,k)

    def test_wwan_network_radio_mode_setandget_2Gmode(self):
        WWAN_System.wwan_network_radio_mode_setandget_2Gmode(self,k)

    def test_wwan_network_radio_mode_setandget_3Gmode(self):
        WWAN_System.wwan_network_radio_mode_setandget_3Gmode(self,k)

    def test_wwan_network_radio_mode_setandget_4Gmode(self):
        WWAN_System.wwan_network_radio_mode_setandget_4Gmode(self,k)

    def test_wwan_network_radio_mode_setandget_23Gmode(self):
        WWAN_System.wwan_network_radio_mode_setandget_23Gmode(self,k)

    def test_wwan_network_radio_mode_setandget_34Gmode(self):
        WWAN_System.wwan_network_radio_mode_setandget_34Gmode(self,k)
```

6、最后的结果:

1）HTML Result



Test Group/Test case	Count	Pass	Fail	Error	View
test_sdk_test	46	46	0	0	Detail
test_AT_Command			pass		
test_get_wwan_serving_system_provider			pass		
test_get_wwan_serving_system_radio_mode			pass		
test_wwan_network_radio_mode_setandget_2Gmode			pass		
test_wwan_network_radio_mode_setandget_3Gmode			pass		
test_wwan_network_radio_mode_setandget_4Gmode			pass		
test_wwan_network_radio_mode_setandget_23Gmode			pass		
test_wwan_network_radio_mode_setandget_34Gmode			pass		
test_wwan_network_radio_mode_setandget_234Gmode			pass		
test_get_wwan_band_capability			pass		
test_get_wwan_serving_system_roaming			pass		

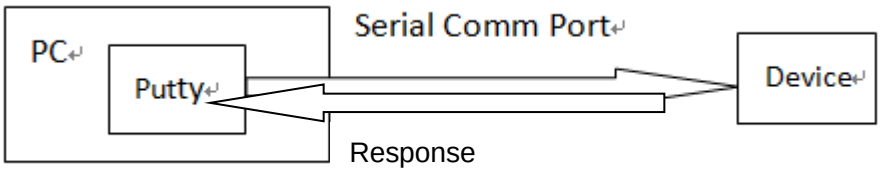
2）Putty 保存的 log: 可以查看测试的 Log

putty_WWAN&Network&System.log - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
{ "action": "get_wwan_serving_system_provider" }
read:
{ "get_wwan_serving_system_provider": { "errno": 0, "errmsg": "", "provider": "UNICOM" } }

root@home:~# JsonClient /tmp/cgi-2-sys get_wwan_serving_system_radio_mode
send:
{ "action": "get_wwan_serving_system_radio_mode" }
read:
{ "get_wwan_serving_system_radio_mode": { "errno": 0, "errmsg": "", "radio_mode": 4 } }

root@home:~# JsonClient /tmp/cgi-2-sys set_wwan_network_radio_mode '{ "radio_mod
e":2}'
send:
{ "action": "set_wwan_network_radio_mode", "args": { "radio_mode": 2 } }
read:
{ "set_wwan_network_radio_mode": { "errno": 0, "errmsg": "" } }

root@home:~# JsonClient /tmp/cgi-2-sys get_wwan_network_radio_mode
send:
{ "action": "get_wwan_network_radio_mode" }
read:
{ "get_wwan_network_radio_mode": { "errno": 0, "errmsg": "", "radio_mode": 2 } }

root@home:~# JsonClient /tmp/cgi-2-sys set_wwan_network_radio_mode '{ "radio_mod
e":3}'
send:
{ "action": "set_wwan_network_radio_mode", "args": { "radio_mode": 3 } }
read:
{ "set_wwan_network_radio_mode": { "errno": 0, "errmsg": "" } }

root@home:~# JsonClient /tmp/cgi-2-sys get_wwan_network_radio_mode
send:
{ "action": "get_wwan_network_radio_mode" }
read:
```

四、AT Command API Automation

1、【待测物分析】测试框架

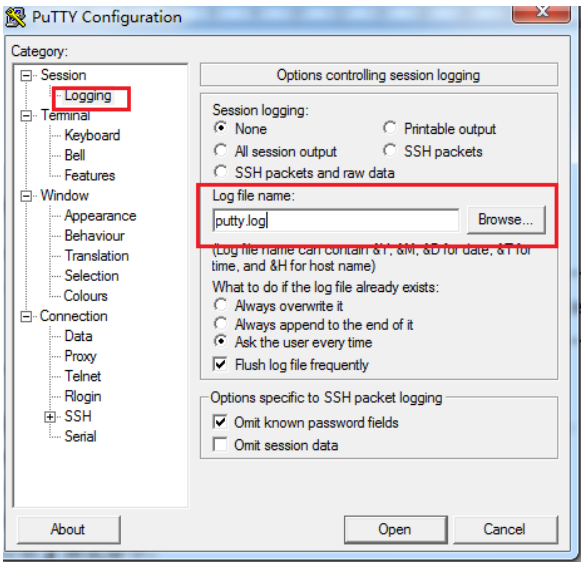


1）通过安装在 PC 上的 Putty 通过 Serial Com port 与 Device 建立连接

2）通过 Putty 向 Device 下命令（Input）

3）Device 给出 Response 并反馈给 Putty （Output）

4）Putty 会自动将 Log 保存到一个设定好的路径，如果保存默认路径则反馈的信息将会保存在与 Putty 此工具在同一个文件夹中。



2、【待测物分析】测试命令

以一个命令为例进行说明:

Input 就是 AT&C0，AT&C1，AT&C2

Response 就是 OK(当然不同的命令会有不同的 response)

2.9. AT+CFUN Functionality Level

+CFUN parameter command syntax

Command	Possible response(s)
+CFUN=[<fun>[,<rst>]]	+CME ERROR: <err>
+CFUN?	+CFUN: <fun> +CME ERROR: <err>
+CFUN=?	+CFUN: (list of supported <fun>),(list of supported <rst>) +CME ERROR: <err>
Reference:	3GPP TS 27.007

The AT command availability is represented in the table below:

[redacted]	[redacted]
Yes	Yes

Example:

Case 1: Query the supported +CFUN values.

```
AT+CFUN=?
+CFUN: (0-1,4-7),(0-1)

OK
```

Case 2: Go to the airplane mode and then go back to normal mode.

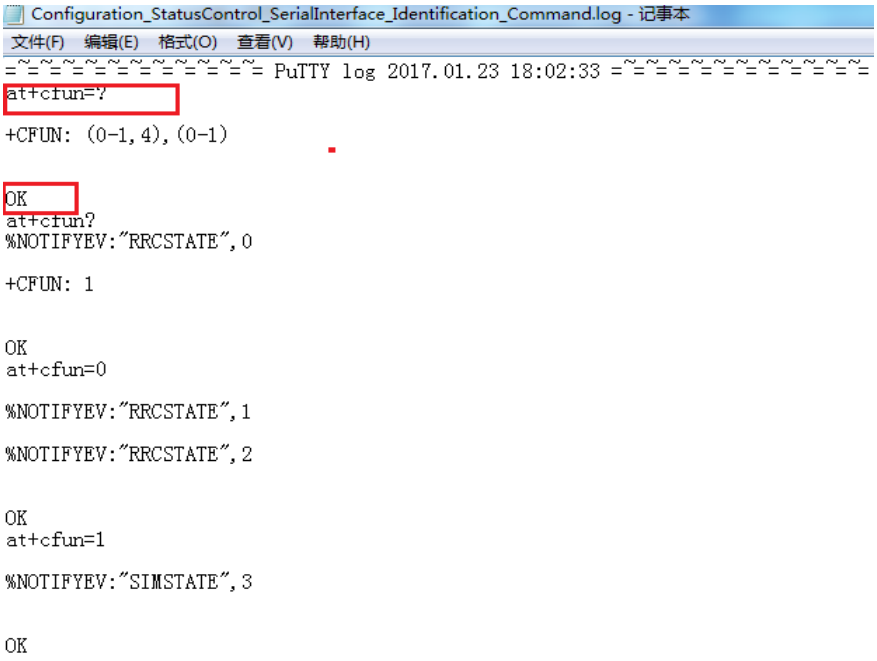
```
AT+CFUN=0
OK
AT+CFUN=1
OK
```

3、【待测物分析】自动化分析

1）测试流程同 Json Command

都是操作 Putty 将 Dos 命令调用出来，然后输入命令，待测物会根据输入的命令返回相应的命令给 Putty。

2）测试完毕后需要到 Putty 设置的路径找到相应的 log，接下来就要在此 putty.log 中提炼出 response，并将此 response 与已知的数值进行对比。



4、Python 的知识回顾

Pywinauto 解决控制 Putty tool 上各种控件的问题：Python 系列课程已解决。

Pykeyboard 解决在类似 dos 界面输入命令的问题：Python 系列课程已解决。

Unittest 解决搭建测试框架，因为测试命令很多，不可能一个 command 一个脚本，如下：Python 系列课程已解决。

2.1. WWAN 7
2.1.1. get_wwan_serving_system_status 7
2.1.2. get_wwan_serving_system_provider 8
2.1.3. get_wwan_serving_system_radio_mode 9
2.1.4. get_wwan_network_radio_mode 9
2.1.5. set_wwan_network_radio_mode 10
2.1.6. get_wwan_radio_info 10
2.1.7. get_wwan_band_capability 11
2.1.8. get_wwan_band_info 12
2.1.9. get_wwan_lte_band_info 12
2.1.10. get_wwan_lte_ca_info 13
2.1.11. get_wwan_serving_system_roaming 14
2.1.12. get_wwan_allow_data_roaming 14
2.1.13. set_wwan_allow_data_roaming 14
2.1.14. get_wwan_network_time 15
2.1.15. get_wwan_ipv4_network_state 15
2.1.16. get_wwan_ipv4_network_ip 16
2.1.17. get_wwan_ipv4_network_connection_time 16
2.1.18. get_wwan_ipv4_network_status 17
2.1.19. get_wwan_ipv6_network_state 18

4）对于 putty.log 的操作：读取指定的位置的字串，并将字串提取出来：此问题需要学习 Python 对于字串的处理。

5）将上一步提取的字串与已知的字串对比根据最后的对比的结果产出一个比较漂亮的报告：Python 系列课程已解决。

[redacted] AT Command Test Result

Start Time: 2017-01-23 18:02:23
Duration: 0:02:16.298000
Status: Pass 5

Test_Report

Show Summary Failed All

Test Group/Test case	Count	Pass	Fail	Error	View
test_atcommand_test	5	5	0	0	Detail
test_at_cfun_help: at+cfun=?			pass		
test_at_cfun_query: at+cfun?			pass		
test_at_cfun_set_0: at+cfun=0			pass		
test_at_cfun_set_1: at+cfun=1			pass		
test_at_cfun_set_4: at+cfun=4			pass		
Total	**5**	**5**	**0**	**0**	

5、Python 对于字串处理的详细分析

1）将 Putty.log 文档打开(Json Command API Automation 此问题已经解决)

2）读取 putty.log 此文档中的信息 (Json Command API Automation 此问题已经解决)

3）从将列表形式的字串转成 String 类型的字符(Json Command API Automation 此问题已经解决)

4）从第三步读取的信息提取希望想要的资料

A、观察命令的反馈结果：我们希望改善 Json Command 字串处理算法的两个问题。

我们需要将查找字串的位置以及建立的 Pattern（希望提高目前算法的效率）的方式。

B、明确我们的目标：定位到该命令的 response 信息的位置，然后根据这个位置提炼出我们希望的字串

观察以下 log，对于 at+cfun=0 这个命令，我们希望提取的 response 是“OK”。

观察以下 log，对于 at+cfun=0 这个命令，我们希望提取的 response 是“OK”。

```
at+cfun=0
%NOTIFYEV:"RRCSTATE",1
%NOTIFYEV:"RRCSTATE",2

OK
at+cfun=1
%NOTIFYEV:"SIMSTATE",3

OK
%NOTIFYEV:"RRCSTATE",0
%NOTIFYEV:"RRCSTATE",1
at+cfun=4
%NOTIFYEV:"RRCSTATE",2

OK
```

C、对应的改善算法：

我们发现 at+cfun=0 这个命令在 log 中是唯一的，所以我们在定位时首先以这个字串为坐标，进行查找字串

注意：自动化命令执行的顺序是：at+cfun=0 ，产生 putty log；抓取此 Log 中的 Response 字串，然后继续下命令 at+cfun=1，产生 putty log；抓取此 Log 中的 response 字串

基于上面的信息，当定位到 at+cfun=0 这个字串后，它后面的“OK”字串是唯一的，所以我们就可以将建立的 Pattern 设为直接查找 OK 这个字串即可，然后使用这个 Pattern 的 findall 函数，查找 at+cfun=0 这个命令后面的字串是否有 OK 即可。

```
def at_cfun_set_0(self,k):
    k.type_string('at+cfun=0')
    time.sleep(2)
    k.tap_key(k.enter_key)
    time.sleep(2)
    f=open(filepath,'r')
    content=f.readlines()
    contentstring=','.join(content)
    #print contentstring
    pattern=re.compile(r'OK')
    start=0
    location=[]
    while True:
        location_index=contentstring.find('at+cfun=0',start)
        if location_index==-1:
            break
        start=location_index+1
        location.append(location_index)
    result=pattern.findall(contentstring,location[0])
    #print len(result)
    #print 'result:'+str(result)
    if len(result)==0:
       search_result=False
    else:
         search_result=response_ok_pass in result[0]
    #print at_cfun_set_0_pass
    #print result[0]
    #print search_result
    self.assertTrue(search_result)
    time.sleep(20)
    f.close()
```

6、实际架构

1）代码整体的组织：

ConfigurationCommand	2017/1/24 11:06	文件夹	
IdentificationCommand	2017/1/24 11:06	文件夹	
NetworkServiceCommand	2017/1/24 11:06	文件夹	
PacketDomainRelatedCommand	2017/1/24 11:06	文件夹	
SerialInterfaceControl	2017/1/24 11:06	文件夹	
StatusControlCommand	2017/1/24 11:06	文件夹	
TestResult	2017/1/24 11:06	文件夹	
atcommandtest.py	2017/1/23 18:05	Python File	23 KB
constant.py	2017/1/23 18:04	Python File	12 KB
constant.pyc	2017/1/23 16:48	Compiled Pytho...	23 KB
HTMLTestRunner.py	2013/7/22 15:54	Python File	24 KB
HTMLTestRunner.pyc	2016/5/16 15:42	Compiled Pytho...	24 KB
putty.exe	2013/12/18 12:16	应用程序	563 KB

2）atcommandtest.py 代码说明

可以理解各个 function 测试的入口，即这里主要是搭建 Unittest 框架

```
#执行测试的类
class test_atcommand_test(unittest.TestCase):
    def setUp(self):
        pass

    #no1.Configuration Command
    #at+cfun=?
    def test_at_cfun_help(self):
        '''at+cfun=?'''
        configurationcommand.at_cfun_help(self,k)

    #at+cfun?
    def test_at_cfun_query(self):
        '''at+cfun?'''
        configurationcommand.at_cfun_query(self,k)

    #at+cfun=0
    def test_at_cfun_set_0(self):
        '''at+cfun=0'''
        configurationcommand.at_cfun_set_0(self,k)

    #at+cfun=1
    def test_at_cfun_set_1(self):
        '''at+cfun=1'''
        configurationcommand.at_cfun_set_1(self,k)

    #at+cfun=4
    def test_at_cfun_set_4(self):
        '''at+cfun=4'''
        configurationcommand.at_cfun_set_4(self,k)


    #statuscontrolcommand
    #at+ceer=?
    def test_at_ceer_help(self):
        '''at+ceer=?'''
```

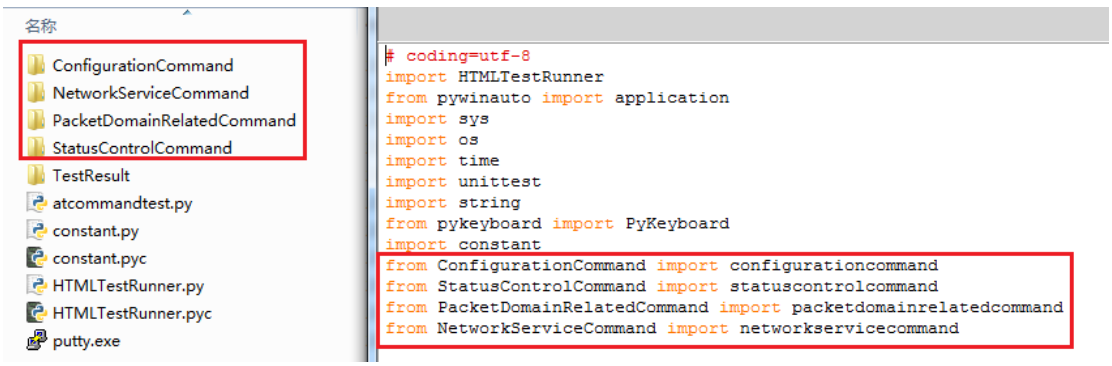
3）constant.py 说明

对于一些经常需要变化的变量，在这里汇集，以函数的方式返回值，为了后续维护方便。

```
import os

#putty path
path=os.getcwd()
putty_path=str(path)+'\\putty.exe'
def get_putty_path():
    return putty_path

#the response of at command is OK
responseok_pass='OK'
def get_responseok_pass():
    return responseok_pass

#the response of at command is ERROR
responseerror_pass="ERROR"
def get_responseerror_pass():
    return responseerror_pass

#configuration command
#at+cfun
#at+cfun=?
atcfun_help_pass='CFUN: (0-1,4),(0-1)'
def get_atcfun_help_pass():
    return atcfun_help_pass

#at+cfun?
atcfun_query_pass='CFUN: 1'
def get_atcfun_query_pass():
    return atcfun_query_pass

#statuscontrolcommand
#at+ceer
#at+ceer
atceer_pass='CEER:'
def get_atceer_pass():
    return atceer_pass
```

4）各个 Function 的脚本的调用逻辑

以一个大块功能一个文件夹的形式存在，而在 atcommandtest.py 此文件夹首先先把各个功能的脚本从这些文件夹中 import 进来，然后才能调用其中的函数。



5）各个 function 脚本的设计逻辑

每个脚本，对于每个命令建立一个函数，此函数后续会在 atcommandtest.py 此脚本中的 unittest 框架中以独立的 test case 被执行。

```
#at+cfun=?
def at_cfun_help(self,k):
    time.sleep(10)
    k.type_string('at+cfun=?')
    time.sleep(2)
    k.tap_key(k.enter_key)
    time.sleep(2)
    f=open(filepath,'r')
    content=f.readlines()
    contentstring=','.join(content)
    #print contentstring
    pattern=re.compile(r'CFUN.*')
    start=0
    location=[]
    while True:
        location_index=contentstring.find('at+cfun=?',start)
        if location_index==-1:
            break
        start=location_index+1
        location.append(location_index)
    result=pattern.findall(contentstring,location[0])
    #print len(result)
    #print 'result:'+str(result)
    if len(result)==0:
       search_result=False
    else:
        search_result=at_cfun_help_pass in result[0]
    #print search_result
    self.assertTrue(search_result)

    time.sleep(2)
    f.close()

#at+cfun?
def at_cfun_query(self,k):
    k.type_string('at+cfun?')
```

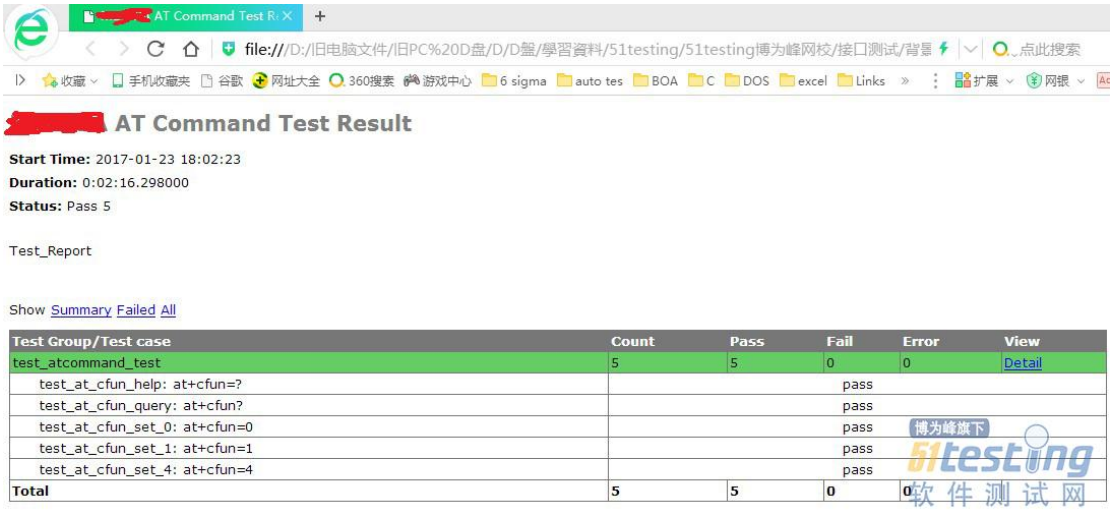
6）与实际 Test case 对比

	Category	Priority	Testing time (Hour)	Remark
Test item 1	Configuration Commands	L1/L2		
Test item 2	Status Control Commands	L1/L2		
Test item 3	Network Service Commands	L1/L2		
Test item 4	Packet Domain Related Commands	L1/L2		

```
if __name__ == "__main__":

    #构造测试集
    testsuite=unittest.TestSuite()
    testsuite.addTest(test_atcommand_test("test_at_cfun_help"))
    testsuite.addTest(test_atcommand_test("test_at_cfun_query"))
    testsuite.addTest(test_atcommand_test("test_at_cfun_set_0"))
    testsuite.addTest(test_atcommand_test("test_at_cfun_set_1"))
    testsuite.addTest(test_atcommand_test("test_at_cfun_set_4"))
    testsuite.addTest(test_atcommand_test("test_at_ceer_help"))
    testsuite.addTest(test_atcommand_test("test_at_ceer"))
    testsuite.addTest(test_atcommand_test("test_at_pdnset_query"))
    testsuite.addTest(test_atcommand_test("test_at_cmatt_setandquery_1"))
    testsuite.addTest(test_atcommand_test("test_at_cmatt_set_0_1"))
    testsuite.addTest(test_atcommand_test("test_at_cpas_help"))
    testsuite.addTest(test_atcommand_test("test_at_cpas_query_default"))
    testsuite.addTest(test_atcommand_test("test_at_cpas_query_4"))
    testsuite.addTest(test_atcommand_test("test_at_cpas_query_5"))
    testsuite.addTest(test_atcommand_test("test_at_cpas_query_4_reconnect"))
    testsuite.addTest(test_atcommand_test("test_at_status_help"))
    testsuite.addTest(test_atcommand_test("test_at_status_set_init"))
    testsuite.addTest(test_atcommand_test("test_at_status_set_rrc"))
    testsuite.addTest(test_atcommand_test("test_at_status_set_usim"))
    testsuite.addTest(test_atcommand_test("test_at_status_set_uicc"))
    testsuite.addTest(test_atcommand_test("test_at_status_set_roam"))
    testsuite.addTest(test_atcommand_test("test_at_cgerep_help"))
    testsuite.addTest(test_atcommand_test("test_at_cgerep"))
    testsuite.addTest(test_atcommand_test("test_at_cgerep_setandquery_0_0"))
    testsuite.addTest(test_atcommand_test("test_at_cgerep_setandquery_0_1"))
```

7、最后的结果:

1) HTML Result



AT Command Test Result

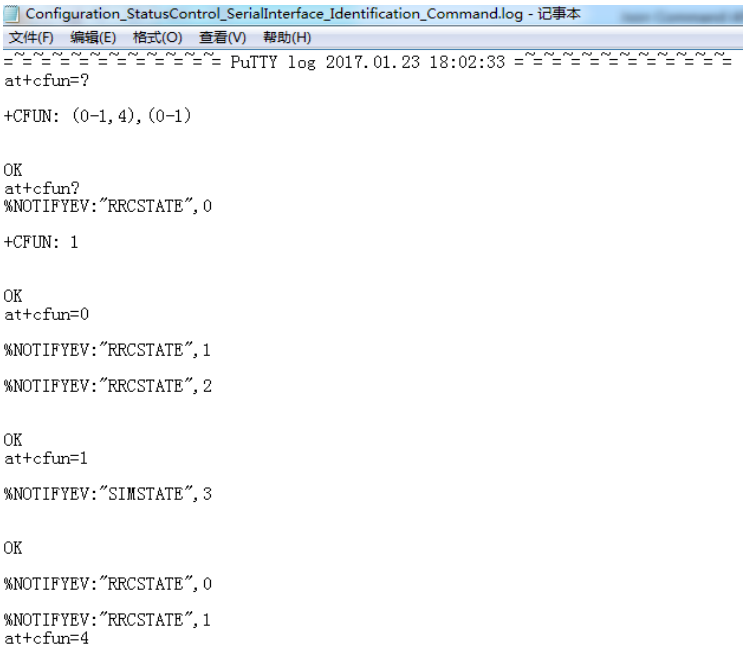
Start Time: 2017-01-23 18:02:23
Duration: 0:02:16.298000
Status: Pass 5

Test_Report

Show Summary Failed All

Test Group/Test case	Count	Pass	Fail	Error	View
test_atcommand_test	5	5	0	0	Detail
test_at_cfun_help: at+cfun=?			pass		
test_at_cfun_query: at+cfun?			pass		
test_at_cfun_set_0: at+cfun=0			pass		
test_at_cfun_set_1: at+cfun=1			pass		
test_at_cfun_set_4: at+cfun=4			pass		
Total	**5**	**5**	**0**	**0**	

2) Putty 保存的 log: 可以查看测试的 Log



Configuration_StatusControl_SerialInterface_Identification_Command.log - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
~=~=~=~=~=~=~=~=~=~=~=~= PuTTY log 2017.01.23 18:02:33 =~=~=~=~=~=~=~=~=~=~=~=~
at+cfun=?

+CFUN: (0-1,4),(0-1)


OK
at+cfun?
%NOTIFYEV:"RRCSTATE",0

+CFUN: 1


OK
at+cfun=0

%NOTIFYEV:"RRCSTATE",1

%NOTIFYEV:"RRCSTATE",2


OK
at+cfun=1

%NOTIFYEV:"SIMSTATE",3


OK

%NOTIFYEV:"RRCSTATE",0

%NOTIFYEV:"RRCSTATE",1
at+cfun=4
```

五、总结

接连几期内容女巫介绍了 Python 的相关知识，我们在实际的工作中遇到了接口测试的实际问题，那么问题来了我们如何用已经学过的知识解决我们遇到的实际问题呢，这次接口测试就是解决这个问题，从上述内容可以看出有的在我们解决实际问题时，会遇到一些之前没学习的新知识，这时就要根据实际需求，去学习需要的新知识。

这里介绍了两种不同类型的接口测试，虽然类型不同但是测试框架，测试手法，测试流程都比较类似，所以解决了一种类型的接口自动化测试，另一个接口测试就迎刃而解。

除了需要学习新知识外，对于我们的测试需求抽象出具体的数学模型，这个能力也非常重要，抽象的这些算法，值得大家不断去思考是否还有更好的算法，更有效率的算法？

我们不停找到工作中可以改善的问题，不停的解决这些问题，最重要的要不停的在总结解决这些问题的方法，，久而久之我们就会觉得没有什么问题是“不可解决的”！

参考文献:

1、Python 官网：http://www.python.org/getit/

2、Pywinauto 的官网 http://pywinauto.googlecode.com/hg/pywinauto/docs/contents.html

互联网产品测试之挑战、工具和测试方法

◆ 译者：侯 峥

互联网产品测试:

当人们被问道生活的必需品是什么的时候，大多数人的答案是：吃、穿、住。

但是，那是一个世纪以前的答案了。人类已经进入到需要为生活发展其他必需品的时代了。我们需要使我们的生活变得更简洁、更好、更方便。

我们已经不需要用开关来开、关灯，不需线下缴费，我们可以更精确的检查自己的身体健康状态，更有效的规划自己的行车路线等等还有很多。

我们现在该怎样做？它是多么的不同？

在我们学习怎样测试互联网产品之前，我们应该了解它。

互联网产品是什么？

互联网产品就是连接汽车、家用电器、嵌入式电子医疗器械、芯片等等，用来收集和交换不同种类的数据的产品。这项技术允许人们通过网络远程的控制设备

互联网产品例子

一些互联网产品在现实生活中应用的例子如下:

#1）可穿戴技术

可穿戴配件像 Fitbit 手环和苹果手表与移动设备连接同步。

这些能够帮助获取必要的信息例如健康、心率检测、睡眠活动等等。这些也可以显示移动设备上的数据和信息。

#2）公共设施和住宅小区

使用一个叫做 CitySense 的应用程序很容易获取户外实时照明数据，基于这些数据路灯可以自定的亮灭。有一些应用程序用来检测交通信号灯和在一个繁华的城市找到可用的停车位。

#3）医疗保健

有很多的应用程序来检测患者的健康状况。

根据基本数据，服务台人员在一天的不同时间段控制药物的用量。像 UroSense 这样的应用程序可以检测患者体内的液体水平，根据需求量设定液体的输入量。同时，数据可以无线发送给相关人员。

互联网产品相关技术

下面列出几个互联网产品相关热门技术

- RFID 标记和 EPC
- NFC 用于电子设备之间的相互通信。是基于智能手机用于非接触式支付交易。
- 蓝牙：用于小范围内通信传输。大多用于可穿戴技术。
- Z-Wave：这是低耗射频通信技术。初步用于智能家居和灯光控制等。
- WiFi：互联网最常用的技术，连上 LAN，可以流畅的传送文件、数据、信息。

互联网产品测试

让我们举一个医疗保健监控系统的例子，这个系统可以监控健康、心率、液体吸收的详细信息并且发送报告给医师。这些数据存储在系统里，可以在任何需要的时候查询历史数据。

医师可以根据这些数据来确定用药量和液体用量，用任何一台连接药物仪器的设备（电脑或移动设备）操作。

现在为了测试这样的仪器，我们需要了解多种测试的方法，比如：

互联网产品测试方法

#1）易用性

我们必须保证每一个使用产品的易用性。

药物医疗保健检测设备必须是便携的，可以被移动到任何的情况下的医疗检测。

设备必须足够智能可以不止发送通知还有错误信息和警告等。

设备应该有日志功能可以提供下一个使用者足够明确的信息。如果没有该功能，系统应该把所有数据存储到数据库中。

通知应该在经过处理后完整的展示在设备上。

展示数据、处理数据、日程代办提醒方面的易用性鼻血被充分的测试。

#2）互联网产品安全

互联网产品安全挑战：互联网产品是数据的核心，所有操作连接的设备/系统处理的原始数据必须是可用的。

当两台设备传输数据时，总有数据传输失败或传输数据不可读取的可能。

基于一个测试标准，我们需要检测数据从一台测试传输到另一台设施是否是受保护的/加密的。

无论怎样，UI 方面要保证密码掩码显示。

#3）连通性

由于是医疗保健设备，连通性 zhiguan 重要。

系统必须在任何时间可以使用，保证相关人员即使了解相关的情况。

至于连通性，有两个方面必须要重点测试。

连通性，当设备连通并且运作时，设备的传输数据和任务接收必须流畅及时。

另外一个条件是使用设备的异常场景。无论系统和网络多么的健壮，系统总是有宕机的可能性。作为一个测试者，我们也需要测试这些异常场景。一旦，系统连不上网，必须提醒医师认为检测病人的健康状况而不是继续以来设备。另外，系统中还需要有易恢复性在宕机的时候可以存储所有的数据。一旦系统恢复工作，所有的数据都会重新上传，在任何情况数据都不能丢失。

#4）性能

当我们讨论的系统涉及到医疗保健领域的时候，我们必须确保系统对于整个医院来说是足够规模的。

当测试结束的时候，系统可以同时供 2~10 个病人使用，数据可以被推送到 10~20 个设备中。

当整个医院连接到一起，180~200 个病人使用系统，被推送的数据要比测试数据多得多。

作为测试者，我们要确保即使推送的数据量加大，系统的性能也要符合要求。

我们也需要测试显示系统使用状态、电量状态、温度等信息的显示设备。

#5）兼容性测试

介于互联网产品使用设备的种类繁多，兼容性测试是必须的。

测试项目如：操作系统兼容性测试、浏览器兼容性测试、移动设备兼容性测试、通信模式【例如：蓝牙 2.0、3.0】都是互联网产品兼容性测试的必要项。

#6）β 测试

对于互联网产品而言，β 测试是必须的。

即便测试环境确保产品/系统是没有 bug 的，然而，在线上环境/步骤/场景还可能有严重的问题。

在 β 测试的时候，选择特定的人员在真时的条件下使用系统。他们使用应用程序并反馈使用结果。

这些反馈有助于更好的改善应用程序。

#7）冒烟测试

医疗保健系统需要通过很多的监管和兼容性测试。

考虑系统在通过所有的步骤但是最后失败的异常测试场景的复杂测试用例。

在系统上线之前最好耐心的去做冒烟测试。这点必须是测试用例中的。

做冒烟测试，我们必须保证系统所有的冒烟测试点通过。

#8）升级测试

互联网产品涉及多种协议、设备、操作系统、固件、硬件、网络等等。

当要升级的时候，确保系统等所有涉及到的项目都考虑到。回退测试和相关的升级

策略必须被考虑到以便应对升级遇到的突发状况。

互联网产品测试的挑战

测试者面临的互联网产品测试的挑战如下:

#1）硬件—软件的融合

互联网产品是一个涉及多种软件硬件的系统。不止软件应用程序使系统运作，硬件、传感器、通信系统也起着至关重要的作用。

仅靠功能测试不能够确保系统完美。还要考虑环境和数据交互等等。所以相较于测试一般的系统【只涉及硬件、软件】，测试互联网产品有些沉闷。

#2）设备交互模式

作为一个涉及多个硬件、软件的系统。在实时传输信息时，信息是被托管的。当系统互相交互时，例如安全、版本兼容、升级测试对于测试团队来说是一个挑战。

#3）实时数据测试

正如我们早前讨论的，对于医疗保健系统做 β 测试和冒烟测试是必不可少的。模拟真是的数据也很困难。

作为测试团队，找到冒烟测试的测试点或者充分做好 β 测试是困难的。如果涉及到医疗保健系统则更加困难。所以，这一点对于测试团队来说更是挑战。

#4）UI

互联网产品涉及的设备 you 很多平台[iOS, Android, Windows, linux]。现在，设备兼容性测试可以做，但是所有的设备都测试到是不可能的。

我们不能忽略没有测试到的设备上的 UI 显示问题，设备兼容性测试对于我们来说是很难克服的。

#5）网络可用性

网络在互联网产品方面的重要性体现在数据传输数率。互联网产品必须在各种网络条件下测试。

为了测试网络，虚拟网络模拟器用于模拟各种强网、弱网、无网等的网络情况。但是，总是有测试者考虑不到的实时数据和网络情况。对于这些方面需要更长时间的探

索。

互联网产品测试工具

测试互联网产品有很多工具，它们可以根据测试的对象分类，如下:

#1）软件

Wireshark: 是一个开源网络封包分析软件。网络封包分析软件的功能是撷取网络封包，并尽可能显示出最为详细的网络封包资料。

Tcpdump: 可以将网络中传送的数据包的"头"完全截获下来提供分析。它支持针对网络层、协议、主机、网络或端口的过滤，并提供 and、or、not 等逻辑语句来帮助你去掉无用的信息。

#2）硬件

JTAG Dongle:是一个简单的电脑软件调试器。它可以调试对象的底层代码并一步一步的显示。

Digital Storage Oscilloscope:是用于检查各种事件的时间戳、电量供应、和信号的交互。

Software Defined Radio:是用于模拟发射和接受大量的无线网关。

随着时代的发展，互联网产品的市场在不断的发展、有更多的前景。在这个发展的世界，互联网产品变成生活必需品只是个时间问题。

互联网产品触发装置，智能设备应用程序和通信模块在研究和评估各种物联网服务的性能和行为发挥至关重要的作用。

糟糕互联网产品触发装置和服务会妨碍应用程序的正确功能，最终影响用户的体验。

总结

互联网产品的测试方法可以根据测试系统/产品的不同而有些差异。测试者应该更多的站在用户的角度而不是仅仅基于需求文档测试。

互联联网产品测试的一个重要测试是集成测试。如果集成测试做的很严谨、健壮，能够测试出系统中的大部分问题，这个互联网产品无疑会是成功的。

互联网产品测试是一个有挑战性的工作。但是，对于测试团队来说，能够测试这样一个有这复杂的设备、协议、操作系统、通信协议、硬件、固件的产品是很让人激动的。

软件测试者需要了解关于自动化的什么?

◆译者：枫 叶

即使你正在使用人工测试，了解关于自动化测试进展如何仍然是重要的。不管你的角色是什么，你的每日工作仍然可能通过使用这篇文章中的至少一些方法被加强。这里，学习一些普通术语的含义和一些他们可能如何被应用在软件开发工厂。

近日来我 Twitter 了关于软件测试者需要了解自动化世界正在发生了什么。它得到了一个漂亮的温暖的回复，所以我想我应该要透露我的想法。

这些天来不管你在测试里的角色是什么，你的每日工作将可能通过以下方法的至少其中一些被强化。从最低程度我建议了解这些术语的含义和他们可能被应用在软件开发工厂的一个例子。

持续的集成服务

过去十年来在软件开发领域到来的自动化一个最大的变化是任务自动化。在过去，像构建一个应用的特殊版本，创建文档，或者更新 bug 报告的状态是人为的。一些团队甚至贡献为了 启动一个版本而负责的“创建人”责任。像这些人为的任务（或者是紧紧地绑定给个人或机器）是消耗时间的，并且创建来为了避免瓶颈，比如创建人占据私人的一天并阻碍新版本被完成。

幸运的是，持续集成（CI）工具通过允许任务被标准化和自动化来挽救。持续集成服务重要地安排和执行任务，一个规则的台式电脑能做的任务并且让这些任务在目标机器上执行而不是它自己。回到创建版本的例子，取代让鲍勃为手工在他的机器上创建版本负责，一个持续集成服务能被集成去选择一个目标机器并且在那台机器上执行版本。不仅使鲍勃不需要身体上在那台版本机器出现，而且能在任意时刻发生版本创建，不管是已安排的或者是为了响应另一个动作。

举个例子，测试者爱丽丝可能想要一个基于最新改变的应用程序版本去看一个程序错误是否被修复，而且她能自己发起版本创建。这个不仅使资源从做代表性任务中自由

运作起来，而且给团队在个人以外和团队流程上给予了更多的控制。你也可以把持续集成任务绑定一起给更深的线程一些任务。学习一个持续集成如何工作是对没有放很多编程的重点在自动化上很好的引子。

使用持续集成的一个途径是跑端到端的测试套装。这些测试经常需要跑数分钟甚至数小时。我使用过持续集成去自旋向上和自旋向下测试机器并且发起在那些测试机器上的测试。相对于在你自己机器上跑这些测试这是一个很大的帮助，因为它允许一个测试开发者当测试到处跑的时候去做其他的工作。持续集成的服务器控制着所有这些任务的方方面面。

一些持续集成服务的普通例子是开源工具 Jenkins，基于云的 Travis CI，和专属工具 Bamboo，但是这些也是其他的一些。甚至更低技术是使用一个像克隆或者 windows 任务分配者的工具为了在单一机器上去使任务自动化。

CI 对于开发软件爱好之外的编程是独立的，并且它是一个测试能确实增加价值的一个地方。

现代源码控制

我首先需要指出我爱源码。当编写代码（或者博客!）时，它是一个很有帮助而不仅是工具。对于一个编码的测试员，它是一个无需脑力者。甚至即使一个测试不编码，当测试软件时以现代方法使用源码控制可能是一个大的利益。

在现代方法中“我”的意思是什么？“我”的意思是使用源码控制 1）集成其他工具，比如 CI 服务器或者问题追踪器，并且 2）允许使用好的团队流程习惯，比如基于干线的开发。好的源码控制允许个人去分析变化和更深地挖掘软件工程正在发生什么。

一个接近源码历史和一些基本培训的测试能问出像“在应用里的哪个文件有最多的开发在它们上面工作？”“哪个文件有最大的变化？”“哪个变化的设置包含引起问题的代码？”等待。这个信息有助于找到步调且暗示一些事件的引发。

用 CI 集成源代码甚至能更加有力。在问题跟踪者的事件能使它们的状态在由开发引起的变化中更新。测试者能要求必要的需求在输入的代码被自动查找出来，比如通过自动测试或者代码模式需求。建构和部署能被改代码发起。当源码控制被很好使用，在这种情况下有很多种可能，这是一个在持续传递后隐含的概念。

举个例子，我在一个使用基于云集成服务的开源项目上工作为了检查每一个由提交

者提交的交付。在这个项目里，持续集成运行所有的自动化测试并且检查所有为形式和格式增加的代码。假如一个提交造成错误的测试，或者没有满足设置的风格向导，提交失败了并且暗示了提交者和项目维持者去修改提交。这有助于提供项目历史里以统一的风格每一个提交并且暗示了提交者在增加或者更新模块中可能的微小错误。

这些目前在源码控制的热点是 Git，自由和开放代码的，在它周边有着健壮的生态系统。这些也是一些其他的方面，比如 Subversion，Mercurial 和微软团队基金会。

遥测和监控

这是一个我并不熟悉的主题，但是它确定是测试者们感兴趣的。监控是一种方法，从此挂钩被放在一个应用程序里去发回关于软件是如何被使用的信息给软件创造者。这能包含正被使用的后端/服务器应用程序接口函数，并且在哪个指令，由被使用的由用户界面组成的部分和在什么频率上，等等。

这个目标不是为了发送特殊的用户信息返回给开发团队，更普通的信息是关于一个应用程序正在被用着的和如何被用的部分。这提供了终端用户在做什么的视角，他们实际上如何使用应用程序，并且特定属性如何被得到。安兰培是个微软测试，曾经简短讨论这事情的他曾做过的通过遥测和监视的一部分。

类似于最小化资源控制历史，监视能帮助你找出答案，从简单的问题中（“上周多少人记录？”）到更特殊的和可视化的问题（“当特性 X 被发布时用户们如何改变他们的习惯？”）。这些是帮助测试们执行更好的测试策略的种类问题，并且，总的说来，帮助团队对用户做更好的选择。

更多的信息，请检查 AB 测试播客页面和布伦特詹森。一个主流产品如何使用遥测技术，看一看 Mozillla 如何通过火狐使用监测技术。

也使用 Selenium

最后一点，但这不意味着这不重要，对于使用 web 应用程序以及其相似的应用程序的测试者来说，Selenium WebDriver 是一个很好的工具。在这一点上，WebDriver 是一个用于自动驱动浏览器行为的标准工具，类似于一个人类用户如何在浏览器中用网站 APP 交互。它有一些语言绑定，和一些主流浏览器工作，并且是一款非常好的能被开发第一组件的可扩展性 API 的例子。简言之，它是一个优秀的工作。

当被灵活地使用时，WebDriver 允许测试和开发去使用户体验性测试得到自动化，

这个可以被放在一个持续性的可传递流程。我写了一个简单的基于网页驱动的测试，可以找到像导航到登录页面的链接的事务，而不是寻找用户名和密码场合（由于坏的部署），或者寻找一个不打开的对话当一个控制被点击成想象的（一个明显的但严重的问题）。这些是很快被找到的事情但是不能被单元测试覆盖。

WebDriver 也能被用在写自动化的测试，可以被本地执行去双重检查那些不会以非预约的方式打断重要特性的变化。这些甚至是 WebDriver 用于扩展功能测试以外的用处。

对于对学习代码感兴趣的测试来说，WebDriver 能提供一个好的学习代码的介绍。自动化测试脚本能是一个容易的方法去熟悉编程而不是深入挖掘代码语言鸿沟。它提供足够的架构去开始，并且仍然能够完成一些很好的测试工作。

大脑有这些概念，加强测试自动化，不管你在软件开发中的角色是什么。

❖ 拓展学习

- 自动化测试工具教程：http://www.atstudy.com/course/explore/Automation

安卓 APP 自动化测试之搞定界面元素

◆ 作者：lamecho 辣么丑

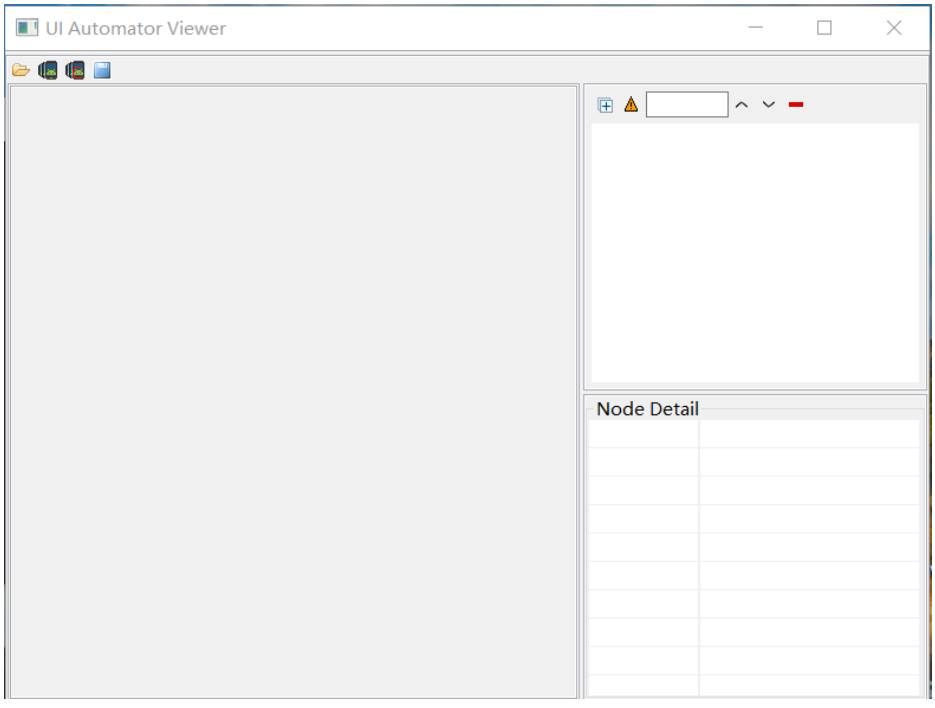
1.1、概要

本篇将对和界面元素相关的内容做讲解，比如，查找元素的几种常见方法，上一篇我们看到的 driver.find_elements_by_id('el')就属于一种；还将介绍如何查看 app 界面上的元素去做定位，最后还将新手经常遇到的问题做个简单的回答。

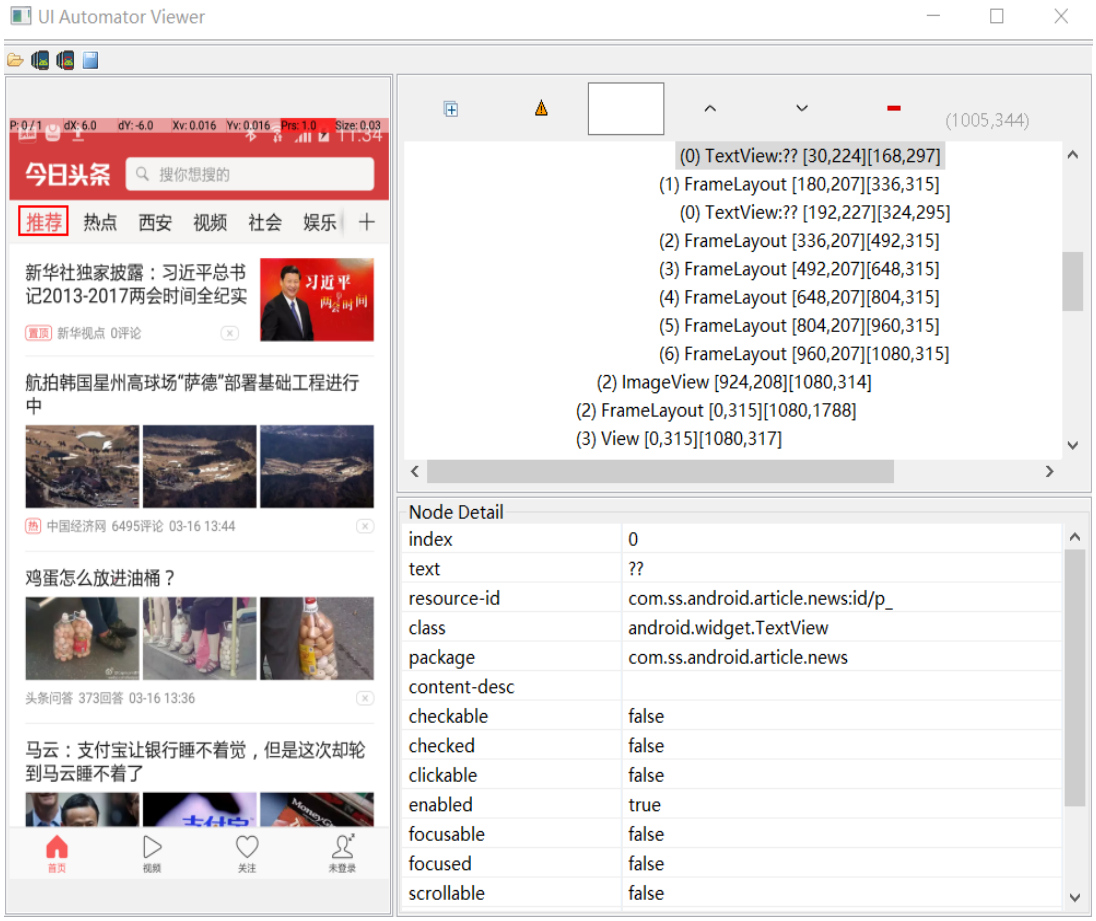
1.2、查找 APP 界面元素的工具

我们先来看看如何找到 app 界面上的元素，以及它有哪些特点需要我们关注的。首先进入我们的 android 的 sdk 文件夹，目录为 C:\Program Files (x86)\Android\android-sdk\tools

在 tools 文件夹下有个 uiautomatorviewer.bat 批处理文件，点开会运行一个叫 UI Automator Viewer 的工具。如图



接下来将我们的手机连接上电脑，启动手机的 app，这里我们继续以今题头条为例子，打开今日头条，然后点击工具左上角的绿色手机图标，等待几秒，结果会出现下图的样子



当我们的鼠标点击到‘推荐’上时，看右侧的信息。主要关注 resource-id，class 这两个标识内容将会是我们后面在脚本里定位元素的主要信息。大家可以先自己熟悉一下这个工具，鼠标在不同的元素上切换看看效果。还记得我们在上一篇脚本中的操作步骤吗？第一步是点击我们的‘热点’按钮将 APP 显示内容切换到热点板块，而脚本里是这样实现的

```
els=driver.find_elements_by_id('p_')
els[1].click()
```

大家看一下我们‘热点’按键的 resource-id 是什么，同样也是 com.ss.android.article.news:id/p_和‘推荐’的 id 一样，同理后面并排的一系列 id 都是一样的。如果简单来说，我们找到了元素并获取到元素的 id 后，我们在脚本里只需要这样写就好了

driver.find_element_by_id('p_')

如果要操作点击

driver.find_element_by_id('p_').click()

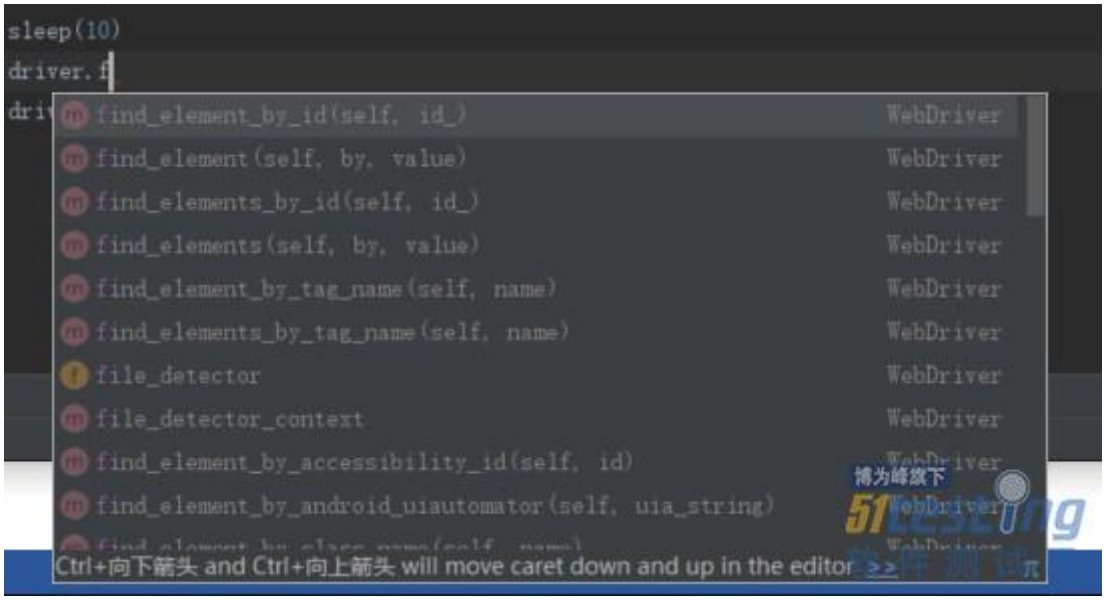
但是，我们的实战碰到的情况是什么呢，今日头条的顶部各个模块的元素 id 都是一样的，这样我们就要根据这个情况，去找 id 都是'p_'的一个集合，当然集合里的第一个元素就是我们的‘推荐’模块，第二个元素就是‘热点’模块，以此类推。所以我们的脚本就要这样去写了

els=driver.find_elements_by_id('p_') #通过 find_elements_by_id 注意是加 s 的

els[1].click()#对 els 集合的第二个元素进行点击操作。

1.3、python 中查找元素的方法

工欲善其事必先利其器，这里先推荐大家一个好用的 IDE—PyCharm。它的智能补全代码功能非常好用，如下图



好了接下来我们来看看 appium 几个常用的找元素的方法:

1、find_element_by_id() / find_elements_by_id()

2、 find_element_by_class_name() / find_elements_by_class_name()

你没看错，不是我漏写了，就这两种。有些童鞋可能要问了，find 开头的方法不是有很多吗？是啊从上面的截图也能看出来，什么 by_tag_name、by_name、by_xpath，by_blabla…

我在这里强调的是常用，好用，实战里用到出现频率最多的就是这两种了。当然在我的下个系列里（学习 selenium 文章），在 web 端查找页面元素的方法就会丰富许多。前面提到的

UI Automator Viewer 让大家注意的两个地方就是 id 和 class 了。在平时在做培训时，经常会遇到这样的问题，很多人会觉得的找元素好难。当每次结束培训后大家会了解难得不是找元素，而是怎么通过编程的思路去在合适的时间找到对应的元素。因为 find 方法是死的，就这几种，为什么别人能找到而我找不到？其实这里展现的问题就是很多人忽略掉的问题本质，你要找的页面元素在你的程序执行所在的界面位置吗？

举个简单的例子，来阐述这个问题的含义：比如 app 的操作是在点击某个界面的一个元素 A 后进入下一个界面，我们要对下一个界面的某个元素 B 再进行点击操作。这时我们的脚本可能是这样写的

```
driver.find_element_by_id(A).click()
driver.find_element_by_id(B).click()
```

大家来看看，这样写我们在执行 python 脚本后会顺利执行吗？答案只有一个 Maybe。也许可能不会出错，但多数情况下会报错，返回无法找到元素 B。这时看出原因的童鞋就要说了在 driver.find_element_by_id(B).click()加上一句 driver.implicitly_wait(10)或是 sleep，脚本就变成这样

```
driver.find_element_by_id(A).click()
driver.implicitly_wait(10)
driver.find_element_by_id(B).click()
```

好了，这里先解释一下 driver.implicitly_wait(10)这句的含义（给初识的筒子普及一下）

这句的含义是程序等待，等待什么呢？就是等待它的下一句里要找的元素 B，直到 B 被找到为止，当然要加一个期限一万年，呵呵开句玩笑，直到 10 秒超时。也就是说在 10 秒内什么时候 B 出现了程序就会开始继续往下执行。当然这里的超时时间可以任意指定，一般我们指定在 10 秒是个比较适合的时间。它被称作“智能等待”，思维活络的童鞋可能就要问了，难道还有非智能的等待吗？当然有了，那就是我们的 time.sleep（）方法，在 import 我们的 time 包后，time.sleep(10)就是实实在在的让程序等上 10 秒才继续往下执行代码。好，到这里说出智能等待的童鞋脸上会露出得意的笑啊，得意的

笑…我这时也有疑问了，这样写我们在执行 python 脚本后会顺利执行吗？答案也只有一个，可以，肯定，必须能顺利执行。但是这里要反转了，虽然脚本能够顺利执行，但是最后执行的结果一定是我们想的哪样吗？答案只有一个 Maybe。为什么？Why？我们不是做了等待吗，脚本不是也顺利找到 B 元素了吗？好了公布答案，回到我们的 app 里，两个界面，A 元素在前一个界面，B 元素在下一个界面，我们脚本里通过 id 找到了 B，那如果说 A 元素所在的界面也有一个和 B 元素同样 id 的元素呢？最后我们脚本的执行结果就是在第一个界面点击 A 后紧接着又点击了当前界面的另一个“B”元素。大家仔细回味一下这个情况，而且此情况会在实战里经常碰到。好了回过头我们看看上一篇我们的那个对今日头条的操作的 demo

```
driver.find_element_by_id('ab0').click()
while 1:
    if driver.current_activity=='com.ss.android.article.base.feature.search.SearchActivity':
        break
driver.find_element_by_id('ll').send_keys('lamecho')
driver.find_element_by_id('o9').click()
```

情况是不是很像，逐条分析一下这几行。在进首页点击了'ab0'后，我们在 while 循环里对当前的界面 activity 做判断，如果获取到当前的 activity 是查找界面的名称，那么我们 break 退出循环，再去找'll'并给他输入“lamecho”字符串，最后点击'o9'搜索。花这么长的篇幅来说明这个情况，就是要说刚才的那句话 “难得不是找元素，而是怎么通过编程的思路去在合适的时间找到对应的元素”。这个时间就是我们在以后的自动化实现上需要去关注和考虑的问题，而找元素那只是自动化必须要有的一个步骤而已。

接下来我们再对 find_elements_by_id()和 find_elements_by_class_name()做个简要的说明。细心的童鞋可能发现了 element 变成了复数的形式。所以通过这种方法找出来的元素必定是一个集合，即返回给我们的是个数组（python 的叫法是个 list）。如果我们的脚本中出现这样的写法你是不能直接使用 send_keys()和 click()的，因为在 python 程序里你不能对一个 list 集合做输入和点击。比如我们这样写:

```
els= driver.find_elements_by_id('ll')
els[0].send_keys(‘lamecho’)
```

对找到的 els 里的第一个元素做输入，第二个就是 els[1]，以此类推。注意下标开始是从 0 开始的。

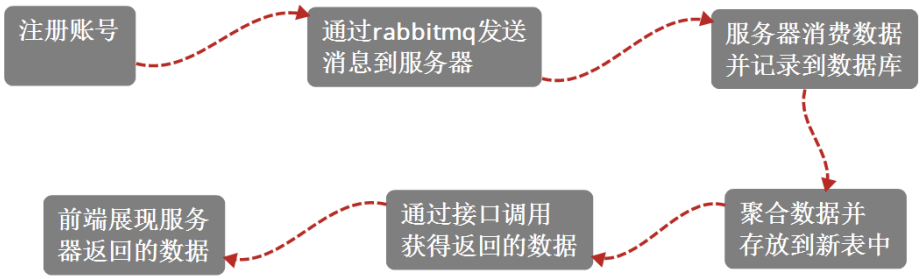
好了，我们的 app 自动化测试第三篇-搞定界面元素也要结束了。希望大家把文中偏于理论的东西分析理解，自己上手再实现一些 demo。下一篇，我们将更多的介绍一下 appium 中有用的一些方法函数，比如说怎么做界面的滑动，怎么做界面坐标位置的点击，锁屏界面的解锁是如何实现的等等。最后感谢大家耐心读完本篇文章！

不可不知的数据分层测试

◆作者：邵君兰

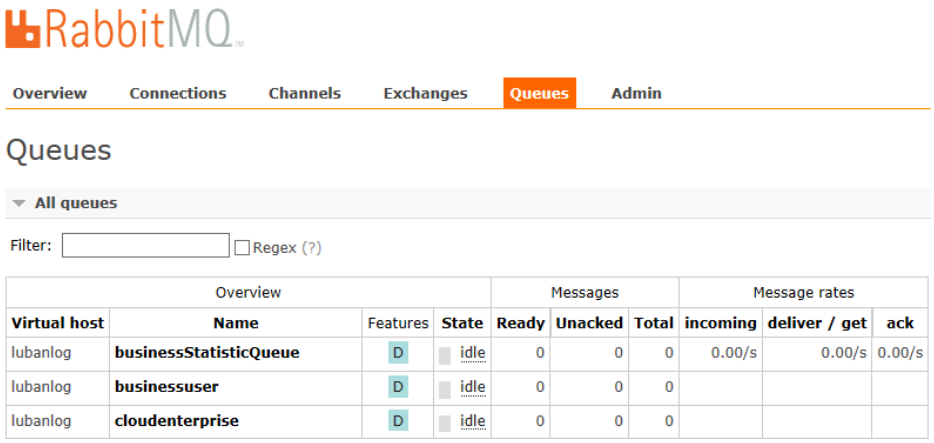
测试效率低下？很多时间都在等程序开发功能，直到界面层展现出来数据后，我们才能介入测试，然后忙的焦头烂额，上线前心里还没底。亦或者发现一个 Bug，发给程序猿 A 查，程序猿 A 说，可能是程序猿 B 的问题，你让程序猿 B 查查。程序猿 B 说，我这正忙着呢，你让 A 先查查，然后这个 Bug 就丢在那里好久，都没有人解决。你除了觉得很苦恼，可能也没有其它办法。如果你遇到这样的问题，那就赶快一起来了解下分层测试吧。

之前有在做企业用户数据统计分析的项目。其中有一个功能，计算新增注册用户的同比百分率。那就拿这个相对简单的功能做实例，讲述分层测试的过程。不过先别着急，在做分层之前，有一件事情是必须要做，那就是必须要了解数据源从哪里来，经过哪些关卡，才最终呈现的页面上来，才能做分层测试。



第一层测试：注册帐号后消息是否进入到 rabbitmq 队列？

如果说消息发送失败，或者在连续发送大量注册信息后队列卡住，那么后面的我们也没有必要去测试了，数据肯定就是错的了。通过下图可以看到消息是否进入队列？是否出现 unacked？消费速度是否快？



当确保这关卡没有问题的时候，我们即可以进入下一关卡的测试。

第二层测试：注册帐号的信息是否按照要求记录到数据库，且无数据丢失？

打开数据库表，查看下数据字段是否齐全，值是否正确？比如说注册帐号要求写入数据库字段有，用户名，注册时间，注册时候需要的手机号码。特别是时间很容易出现 bug，如果要求记录到秒，但是却记录到分钟或者小时，都有可能造成后续数据上的统计错误。

再核对下第一层测试中发送的数据量和写入数据库的数据量是否对等，如果有多了，有可能是数据重复发了，少了可能是数据写入有问题。

当确保这关卡没有问题的时候，我们即可以及进入下一关卡的测试。

第三层测试：聚合数据程序是否正确？

考虑到统计性能，往往对原始数据做一次聚合，减少数据量，提升统计性能。比如说要求是将注册的账号数量按月聚合存放到数据库表中。那么我们就需要去验证聚合后的数据库中的数据是否正确。

首先将原始数据库表中的数据通过查询语句把某个月的数据 count 出来，再到聚合后的数据库表中去核对该月的数据和 count 出来的数据是否一致?假如这块测试下来也没有问题，那么我们就妥妥的进入下一层。

第四层测试：通过接口调用，验证服务器返回的数据是否正确？

虽然经过上一层的数据库测试后，存放的数据是对了，但是服务器返回的数据可不一定正确。因为服务器可能会拿数据库的数据再次做运算后返回给前端。

比如说同比数据，就需要再次做运算，服务器把当前月份的注册用户数据（A）和

去年同月份注册用户数据（B）取出来，然后通过(A-B)/B*100%的运算得到同比数据后返回给前端。可能有的同学会觉得这个运算那么简单，是否可以跳过？有意思的是，程序猿就在做这个算法的地方计算成了(B-A)/B*100%,导致了一个 bug。此时你不用再去确认，是不是发送数据出问题了，还是说数据存放有问题，还是说聚合有问题，直接召唤写这段代码的程序大哥，嘿，就是你那里出问题了，并告知如何修改。是不是很有成就感?至于接口调用测试，你可以借助接口测试工具，如 SoapUI，Jmeter，或者 python 都可以做。

第五层测试：前端页面展示是否正确？

终于测试到最后一层了，经过前面的层层把关，到这里可以说 bug 不会太多了，除非前端代码太烂。那么到这层我们测试些什么呢？比如说前端发送的请求是否正确，假使我要查询 2 月同比数据，愣是发成了请求 1 月的同比数据，那数据肯定又是错了。可能会有人问题，接口测试不是第四层做过了吗？怎么这里还要做？

很简单，第四层我是通过接口测试工具自己写的请求体，参数传的正确，服务器才能返回正确，但是前端程序猿在写调接口参数时，我可不保证他能把请求传正确，所以这是个很重要的验证点。通过抓包工具 fiddler，httpdebugger 都可以轻松找出错误。

其次就是数据在前面的展现是否正确，这块按着方案来测试基本也就没有什么太大问题了。往往数据显示保留位数，数据过长显示不全或者换行等地方会出现问题。

现在有没有体会到分层的好处了？整个测试过程思路始终是清晰的，每一个环节都能层层把关。我不需要等前端做好了，才能测试，任何一个环节做好了，我都能测试。等等，我估计有人又会有疑惑了，这个五层不是一环扣一环的吗？是不是一定要按这种顺序来测试？假设程序猿提前将第三层，聚合数据这块做完了，其他的第一层和第二层都没有完成，我能不能测试？

答案是肯定的，虽然我们缺少第二层中的用户注册数据，但是可以通过写脚本，将注册数据插入到数据库，再跑一下聚合程序，就可以测试了。

掌握分层测试的方法，能缩短整个项目的周期，且质量有保障。程序开发过程，测试也一并在测试，时间并行后，周期就缩短。另外刚写好的热乎代码，马上测试，发现的 bug 也能更容易找到并能立马改掉，程序猿效率也提升了。改变测试任务前松后紧的状态，你值得拥有。

让我们成为开源软件测试者

◆ 作者：孙 远

摘要

随着开源软件的兴起，其使用范围也越来越广泛。开源软件本着“不要重复造轮子”的原则，与商业软件相比，拥有使用成本低、可定制性高等特点。然而开源软件在质量保证上仍然存在诸多的问题。本文将从开源测试工具、开源社区测试能力短板等方面来展示测试人员参与开源社区贡献的必要性。

1、开源测试工具概况

测试工具是质量保障中的“武器”，而测试工具的设计者则是精良武器的“兵工厂”。利用优秀的测试工具可以提高测试效率，发现隐藏较深的软件缺陷。根据测试工具的功能，我们将其划分为测试管理、缺陷管理、持续集成、功能测试、性能测试、测试框架、测试设计、安全测试等类别。下面列举了这些分类中一些典型的测试工具。

- 测试管理：TestLink，Testopia
- 缺陷管理：Redmine，Bugzilla，Mantis
- 持续集成：Jenkins，Buildbot
- 功能测试：Selenium，LTP (Linux Test Project)
- 性能测试：lmbench，Sysbench，Iperf，Fio
- 测试框架：JUnit，Autotest
- 测试设计：Xmind，StarUML，UML Designer
- 安全测试：Metasploit，Nessus，AppScan

在新的开源测试工具不断涌现的同时，很多曾经优秀的开源测试工具因为缺少必要

的维护而无法持续进行新功能的添加和已有功能的增强与维护。Lmbench 是一款优秀的 linux 性能测试工具，可以获取系统 CPU、内存、磁盘、网络等资源的带宽和延时的性能数据。近年来由于缺少必要的官方维护，没有新功能添加，该工具在行业内的影响力在逐步下降。另一个例子是 Rth，其是一款优秀的测试需求管理、测试过程跟踪的测试工具，由于多年来缺少必要的维护，该工具已经无法满足当前项目的需求，逐步淡出了大家的视野。早先原本使用该工具的程序员，不得不将目光转移到其他工具中，而熟悉新的工具需要额外的时间投入，增大了项目运作的成本。

2、当前开源社区测试能力的短板

本章我们以 Kernel 和 Docker 社区为例展示当前主流开源社区的质量保证中有待改进的方面。

2.1、Kernel 开源社区

Linux kernel 是迄今为止最为成功的开源社区，有很多一流的公司和程序员参与其中，平均每月的代码补丁合入量在一千以上，社区的代码量在千万行的数量级。面对如此庞大的代码量，做好质量保障工作是一个严峻的挑战。而目前社区没有精准测试方案，修改一行开发代码时不得不运行全量的测试用例才能充分保证质量。

操作系统运行分为内核态和用户态。当前内核态的测试用例和功能代码存放在一个 Git 仓库中，代码路径在 kernel/tools/testing/selftests 中。而用户态的测试用例与开发代码相对独立，单独存放在 Ltp 开源社区中。

Kernel 社区采用的是迭代式开发。不幸的是，很多开发者在提交完功能代码后并没有同时添加文档和对应的测试用例。而社区也没有一种有效的机制保证文档和测试用例得到及时的更新。文档的输出相对滞后，使得很多内核的特性难以理解，很多时候其他程序员不得不通过阅读源码来推测该代码所实现的功能。而测试用例的滞后输出更为明显，很多内核特性在功能代码合入后的几个月甚至一年以上的时间都没有对应的测试用例。以 user namespace 内核特性为例，其是在 2013 年 2 月 18 日随内核 3.8 版本正式发布的，然而直到 2015 年 5 月 21 日，社区才拥有第一个该特性的测试用例。二者时间间隔在两年以上，也就是说，在这两年中使用该内核版本的用户都无法获取到有效的测试用例来验证该特性，版本的质量保证令人堪忧。社区中有很多类似的待测点，需要大家参与社区来补充必要的测试用例。

另外 Kernel 还维护着 LTS 的长期稳定版本，即在某一个特定的 Kernel 版本中不添加新的功能，只修改已有的软件缺陷。据笔者所知，因为测试资源的缺乏，在 LTS 版本发布前，并不是每个版本都进行源码编译、全量测试执行等测试工作。很多时候验证工作往往只包含源码编译和 OS 启动冒烟测试。这很可能导致大量软件缺陷遗漏到下游开发者手中。

2.2、Docker 开源社区

Docker 是当前流行的云计算开源工具，其社区也吸引了大量的开发者参与社区贡献。Docker 社区采用的是测试驱动开发的策略，即在开发者提交功能代码时需要同时提交对应的测试代码和进行文档的修改。这种开发方式很大程度上解决了文档和测试用例滞后的问题。然而，Docker 社区中并没有专职的测试者，测试用例几乎都是开发者提供的，他们更关注开发代码的质量，对于测试代码的质量就显得有点“漠不关心”了。用例在设计过程中往往缺少测试思维，使得输出的测试用例缺少边界异常点检查。开发者输出的用例几乎都是单点的功能验证，无法覆盖全面的代码路径，更缺少一些专项测试（性能测试、压力测试、长稳测试、安全测试等）。

很多开发者通常是为了应对测试驱动开发的策略被动的添加测试用例，很多重要的特性只使用有限的用例来进行验证。这很难保证软件发布后的质量。在近期发布的 Docker1.12 版本中，就出现了软件稳定性差的问题，引发了很多争论，社区又不得不亡羊补牢。

3、专业测试人员参与开源社区贡献的必要性

从心理学的角度来看，开发者不愿意看到自己编写的代码存在缺陷，也就没有足够的动力去发现软件缺陷。在版本发布临近、任务紧急的情况下，当开发者本人发现了自己编码中的缺陷，而解决该缺陷又需要大量时间并且自己不得不加班时，开发者有两种方式应对。第一种是积极的投入时间，加班加点把问题解决掉来确保版本发布的质量；另一种是保持沉默，默默的祈祷其他人不要发现这个缺陷。经过调查，绝大部分人会选择第二种方式，因为人性即如此，我们做任何事的时候都不要去轻易的挑战人性。而查找软件缺陷是测试者的本职工作，他们会尽最大可能的发现软件缺陷。一旦缺陷被发现，测试者绝不会保持沉默，这有助于最大限度的暴露软件缺陷。 除此之外，开发者设计的用例往往只限定在单元测试中，而集成测试、系统测试、验收测试往往需要测试者的参与。一些专项测试（性能测试、压力测试、长稳测试、安全测试等）也是测试者

所擅长的，他们会站在用户的角度上思考软件的可用性、用户体验等方面的内容。

4.测试工程师参与开源社区的方式

(1)添加测试用例

测试用例的编写是测试工程师的本职工作。很多开源社区非常愿意吸收新的测试用例。在 Docker 社区中，相比于开发代码，测试代码更易于被社区接收。

(2)书写或补充文档

很多社区都因为缺少必要的文档，导致用户无法正确使用软件。开发者书写的文档往往会出现跳步和不准确的情况，需要测试者对文档的书写进行校正。

(3)修改软件缺陷或添加新功能

测试者也可以修复软件缺陷和添加测试所需的新功能，这包含一些可测试性的功能开发。

(4)测试待发布版本

测试待发布的版本来保证软件发布质量，这是测试者最重要的工作职责。

(5)补充专项测试方案

补充一些专项测试（性能测试、压力测试、长稳测试、安全测试等)来完善质量保证体系。

(6)提交软件缺陷和回答社区中的问题

社区往往会通过 issue 列表或邮件对已有问题进行讨论，测试可以提交缺陷或者回答其他人的提问来与社区进行互动。

(7)设计开源测试工具

测试工具往往存在于某个独立的开源项目中，参与这类社区的人中测试者居多。

5、新人参与开源社区的步骤

对于未曾参与过开源社区的新人，我们以向某个开源项目提交测试用例为例介绍参与开源社区的步骤。

首先，你需要熟悉开源代码维护工具，如 Git。掌握代码下载、提交、分支创建的

命令。之后要熟悉被测软件的功能，开始时可以以用户的方式来使用开源软件，掌握软件的使用方法。然后进一步熟悉软件的架构、实现原理、功能函数。再之后的任务是对社区已有的测试用例进行源码阅读，熟悉其测试的功能，寻找到已有测试用例中未覆盖到的软件功能点。针对这些功能点，就可以提交对应的测试用例。

不同的社区接收代码贡献的方式有所不同，Docker 社区是通过 pull request 的方式接收新代码的，而 Kernel 社区则是以 mail list 的方式接收新代码。社区通常会有参与社区的引导文档来引导贡献者，读者在提交代码前可先阅读此类文档。

总结

贡献开源社区的方式有很多，读者可以根据自己的兴趣有针对性的投入到社区的贡献中。随着越来越多的人投入到社区贡献中，相信开源社区的质量保证必将拥有一个质的飞跃。

小白们需知的测试流程

◆作者：兔子闹

不管你是刚进入测试行业还是准备进军测试行业，不管你是在测试部门健全的公司还是测试部门刚建立的公司，最起码你需要了解测试的整个流程。这样你在工作中才能做到心中有数，不至于太慌乱、进去之后四处抓瞎。

其实在进入这个行业的时候，自己也有过迷茫，看起来简单的事做起来却并不简单。甚至很多时候自己觉得自己无事可做，因为在小公司任职，老板也无暇顾及测试整天都在忙些什么，自己也真的是整天悠闲悠闲的混日子，到了项目上线时自然忙的焦头烂额、心里也很没谱，后来跟一些测试大神沟通完之后，真是获益匪浅。那么我总结了下从整体上如何安排测试的，或许并不是绝对的完善，但是希望给那些跟我一样有过迷茫的孩子一点参考。那就言归正传吧。

首先，你需要知道的是，项目立项开始，是否有《需求说明文档》（也就是《需求说明书》）？

如果有，那么前期以此为准开展评审，再审以及最终确认。由于需求评审阶段各个部门及人员所采集的信息不同，但最终都必须达成一致，如果需求确认后，不同部门的工作内容也就明确，开发来说主要确定开发语言，数据库，同时需要输出概要设计和详细设计文档（包括功能的概述，逻辑的整理，建模等），一般服务器等配置这个阶段也可以开始着手做，产品需要根据需求设计原型图及 demo。测试在此阶段需要做的就是确认需求是否都是明确的，是否都是可测的，逻辑是否都是合理的等等，应及时反馈需求中存在缺陷，作为测试负责人需要做的就是工作的记录用时多少个工作日，都干了什么？得到什么结果？是否有遗漏？组员是否都参与？是否对过程中的输出项已经熟悉并认同等等，在公司大流程以及部门规范前提下，当前的工作是否已经可以结束并进入下一阶段？和其他部门是否达成共识？有不明确项要及时发起会议，会议过程中要做好会议记录，会议结束发送参会人员确认，同时明确分工，截止时间等，目前公司是 team leader 和主管去，谁去无所谓，重要的信息的共享和及时传达，人家带你就去，不带就算了，他们得给你详细文档，如果没有文档的话，就功能点细节你得多交流，刚才说那

么多等到该确认的都已确认，作为负责人来说接下来就该准备测试计划了，编写出完整的测试计划，有测试覆盖面，人员安排，工作评估，异常事件处理，结果总结，文档交付，组织评审及最终确认，测试人员就按着测试负责人的计划来进行工作，测试用例的编写（一般都有模版），测试环境的准备，缺陷管理规范和缺陷工具等，理论上用例也需要发起会议评审，分内部和外部直到最终确认，以上都是你有了《需求说明文档》之后，以此为基准开展的前期的测试准备工作，也就是未接到测试版本之前需要做的，理论上测试计划不怎么可靠，因为过程中发生什么谁也无法预料，但是好的计划前期的确可以说明一些事，无非就是设备调用，人员安排，测试覆盖面和深度以及一些特殊情况说明，切记所有的信息的传递及确认一切以邮件为准，为了避免日后的麻烦千万不要认同口头答复。

接下来就是接收版本开始测试，这时候前阶段的准备就可以发挥作用了。作为测试负责人，接收测试版本后可以安排测试，首先就是冒烟测试，验证当前版本是否满足后续测试，满足则发邮件说明，不满足也同样发邮件说明并打回该版本，这个过程基本是已经规范好的，没什么特别说明。

好了，说完有《需求说明文档》的公司，下面说没有《需求说明文档》的公司。

目前大多数公司都这样，在这类公司工作说白了就是干活，别跟人家提流程、文档和规范，人家要么不了解、要么压根就不知道，第一，时间不允许，没空搞那些，既然答应多久能给出产品，那么为了能在相应的时间内完成产品，一般情况下需求人员也是刚弄明白需求，就召集开发人员进行口头讲解，然后开发人员就直接进行开发，若在开发中有不明确的，就再来问需求人员，边开发便询问需求人员；第二，怕麻烦，几句话能讲清楚的事何必麻烦的进行文档的编辑，万一需求变动大，那岂不是三天两头进行需求文档的编辑，需求变动大又能怎样，这时候需求人员又从新讲解，开发人员又接着从新编写代码；第三，老板及管理层只看结果，测试的存在性极低，存在也是为别的部门背锅。以上的情况决定了测试获取信息的不全面，受阻，这样的情况可能大多数人会选择离职或者破罐子破摔，每种环境的存在都有其合理性，当然这样的环境也可以锻炼人，为了自保，工作还得认真的做，他们乱自己不能乱，信息获取不全面可以直接去问，不要怕麻烦，沟通也要有技巧，不要撕破脸（也可以撕破脸，那就是你准备要离开的时候）。有测试任务，先确认任务的目的，内容以及异常情况，一切搞清楚发邮件，来编写测试计划，说明测试时长，测试覆盖范围，异常因素影响测试的等等。

这里就有个很大的问题就是这个测试计划的安排与测试时长，因为前期我们并不了解需求，这样就不能很好的了解每个模块的难易程度也就无法准确安排自己的工作。那肯定有人说，那在需求人员给开发讲解需求的时候你也去旁听就好，我觉得这个方法也可以，但是最大的问题是你一个人还是无法了解整个系统的需求，除非有多少需求人员就有多少测试人员，一个测试跟一个需求人员，想必这个也很难实现，因为公司宁愿多招几个需求和开发也不愿多招一个测试，就是我之前说的，因为老板觉得测试闲暇时间实在是太多、忙的时间太少。然而并不是测试不想忙，只是在前期他们觉得测试人员没必要参与。测试人员在拿到测试版本之后才开始慢慢了解这个系统，但是，完全没有多余的时间去深入了解这个系统，因为这个时候离上线时间不远了，得抓紧测 bug 了，改完了好去上线。看吧，这样测试的作用就是看功能是否报错，界面过的去就行。老板对测试的成功检验也只是功能是否报错，如果上线时没有错误那就是你测试测得好，有了问题就是你测试没好好干活，他忽略了一点就是测试只是尽可能的降低错误，是人就会犯错，更何况是人写的代码，牵一发而动全身的代码，不是改了错误就没了，也可能会迸发其他错误。出了问题测试永远是最大的背锅者，但是没办法，你干的就是这一行，而行业中大多数公司也都是这样，要想在这种环境下生存，你唯有适应这个环境，尽可能的完善自己，多学点知识充实自己，在工作中，大方针不变的情况下灵活点，结合自己的实际情况稍微变通自己的工作方式。

正常进行后续测试过程中想必很多人都会遇到这几类问题：第一，缺陷的提交五花八门，如果有好的缺陷规范此类问题就可避免。第二，提交的缺陷开发不认可，作为开发肯定觉得我的程序怎么会出问题？这时候可以找开发复现，自己提交的缺陷尽可能做到证据确凿（文字搭配图片，必要情况下可以录制下来），用事实说话，言语上不要争执，如果还是不认可，反馈给开发负责人，让开负责人来进行判决。第三，测试部门内部人员的意识和技术培养（一个团队就是一个整体，不要有个人英雄主义，技术只有分享了才是技术），开始测试后这个过程大致都一样，无非就是回归再回归

作为负责人做好下面几件事:

1、内部工作安排，信息收集，结果汇总，异常情况处理

2、全局把控，缺陷预警及反馈

3、与项目组其他负责人的沟通和协调。

因为我所在的公司是一个业务导向性公司，我也只是针对我所在的公司的一些感想，可能不是很全面，所以，再次强调，这篇文章也只是仅供参考，大家还是应该结合自身实际展开工作。

拓展学习

- 系统测试流程概述：http://www.atstudy.com/course/99
- 软件需求精讲：http://www.atstudy.com/course/113

运用场景覆盖管理提升测试精细化管理的实践与展望

◆ 作者：王善民

一、背景

传统的测试管理模式有几个特点：以案例为核心，基于案例的勾选进行测试范围管理和进度管理；以进度为基础，通过测试问题数偏差进行质量管理；以测试经验为手段，凭借测试经理经验，来评估测试的深度和广度。

在测试管理中，下面的场景是否似曾相识？

场景一：

组长：“这个项目 A 的测试进度为什么滞后啊？”

项目名称	计划覆盖率	实际覆盖率	覆盖率偏差	计划通过率	实际通过率	通过率偏差	受理问题数	确认缺陷数	测试经理	规模（人日）
项目A	61.6%	51.3%	-10.3%	58.1%	50.2%	-7.9%	20	16	比尔	654
项目B	61.6%	53.8%	-7.8%	58.1%	52.1%	-6.0%	32	21	伯格	875
项目C	61.6%	61.1%	-0.5%	58.1%	61.1%	3.0%	8	6	大卫	188
项目D	61.6%	61.8%	0.6%	58.1%	61.8%	3.6%	12	7	汤姆	245
项目C	61.6%	62.3%	0.7%	58.1%	61.6%	3.4%	79	61	杰瑞	1860

测试经理：“呃，测试案例漏执行了，我现在就让大家去执行案例去。”

场景二：

组长：“这个项目测试进度怎么样了？”

测试经理：“测完了，进度 100%，案例全部覆盖!”

组长：“那看下这个验收测试问题是什么情况？”

测试经理：“(⊙o⊙)哦，测试案例有啊，也勾了啊，难道测试人员漏执行了？或者是误勾选了？”

	序号	案例编号	案例名称	批次	案例类型	执行方式	执行结果
	1	GT_Z2...	SCMS案...	无批次	功能测试	手工测...	通过
	2	GT_Z2...	SCMS案...	无批次	功能测试	手工测...	通过

从上面两个场景可以发现，传统的测试管理模式存在一些问题。测试进度以测试人员手工执行案例数来体现，存在主观因素导致覆盖偏差，比如没有按时执行案例、误勾选案例等。由于存在一些因素导致测试进度本身有偏差，而以进度为基础的质量管理也会产生偏差，无法真实的反映出测试过程中的质量风险。经验丰富的测试经理可能会对项目管理的好一些，而经验不足的测试经理可能无法掌控项目测试过程，无法识别过程中的质量风险，导致项目交付质量较差。因此传统测试管理对测试经理的经验要求极高，存在个体偏差，依赖于人治，不具推广性。

为了解决传统测试管理存在的不足，我们引入场景覆盖管理，提升测试质量管理的精细化。

一、场景覆盖管理介绍

场景覆盖管理是以场景法为理论指导，梳理测试场景，并通过技术 SQL 统计场景覆盖情况的一种通用的技术管理方法。场景覆盖管理包含测试场景编制，测试场景 SQL 编写和统计监控分析三个步骤组成。

1、测试场景编制

在需求讨论阶段，测试人员根据业务需求场景编写测试场景，并细化为场景案例；在详细设计阶段，测试经理组织开发骨干和测试骨干对场景进行补充评审，确保场景的完整性。下图为场景示例。

项目编号	项目 场景汇总 清理	应用 批量替换	部署环境	场景序号	开发人员	测试人员	业务层	模块层	第一层场景	第二层场景
CC20161234	金融市场推广项目	F-WMO	4月份版本对公	190006	王彬	李君	跨行发二代	落地处理	放行	人行回应失败
CC20161234	金融市场推广项目	F-WMO	4月份版本对公	190007	王彬	李君	跨行发二代	落地处理	不放行	
CC20161234	金融市场推广项目	F-WMO	4月份版本对公	190010	王彬	李君	跨行发二代	WMO发起撤销	落地待处理	
CC20161234	金融市场推广项目	F-WMO	4月份版本对公	190013	王彬	李君	跨行发A112报文	直接发报	发送成功	人行回应成功
CC20161234	金融市场推广项目	F-WMO	4月份版本对公	190014	王彬	李君	跨行发A112报文	直接发报	发送成功	人行回应失败
CC20161234	金融市场推广项目	F-WMO	4月份版本对公	190017	王彬	李君	跨行发A112报文	落地处理	放行	人行回应成功

2、测试场景 SQL 编写

在编码阶段，组织开发人员根据场景编写场景统计的技术 SQL，并安排技术 SQL 的集中评审、试运行，提高场景的准确性。把场景和 SQL 纳入场景监控工具集中统一管理，并把统计日期、多日志等因子参数化，使 SQL 编写更灵活通用，复用度高。下图为

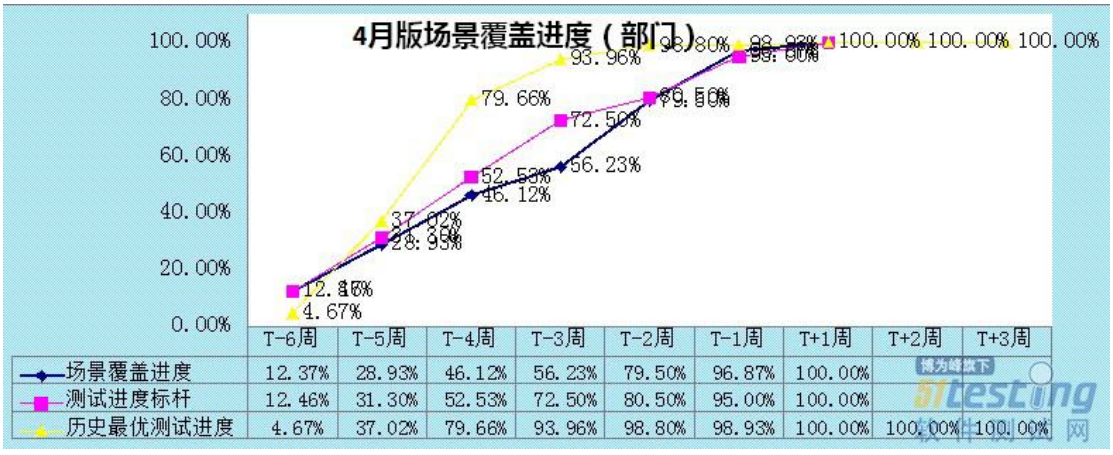
补充监控 SQL 语句后的示例。

项目编号	项目 场景汇总 清理	主应用 批量替换	部署环境	场景序号	开发人员	测试人员	业务层	模块层	第一层场景	第二层场	sql语句 手工确认 手工复制
CC20161234	金融市场推广项目	F-WMO	4月份版本主机对公	190006	王彬	李君	跨行发二代hvps.111报文	落地处理	放行	人行回应	SELECT COUNT(*) FROM NGDBA.WTHIFDT WHERE BUSITYP = 21 AND DRCRF2 = 2 AND MATID = 'HVPS.111.001.01' AND DLSTATUS = 13 AND PROCFLAG <> 8
CC20161234	金融市场推广项目	F-WMO	4月份版本主	190007	王彬	李君	跨行发二代	落地处理	不放行		SELECT COUNT(*)
CC20161234	金融市场推广项目	F-WMO	4月份版本主	190010	王彬	李君	跨行发二代	WMO发起撤销	落地待处理		SELECT COUNT(*)
CC20161234	金融市场推广项目	F-WMO	4月份版本主	190013	王彬	李君	跨行发A112报文	直接发报	发送成功	人行回应	SELECT COUNT(*)
CC20161234	金融市场推广项目	F-WMO	4月份版本主	190014	王彬	李君	跨行发A112报文	直接发报	发送成功	人行回应	SELECT COUNT(*)
CC20161234	金融市场推广项目	F-WMO	4月份版本主	190017	王彬	李君	跨行发A112报文	落地处理	放行	人行回应	SELECT COUNT(*)

3、统计监控分析

在功能和验收测试测试阶段，通过场景监控工具每天自动运行技术 SQl，统计场景的覆盖情况。按周期展现覆盖结果，根据部门、开发组、测试组、项目应用等不同维度生成报表展现，以此为基础分析测试进度、深度与广度，评价测试情况，识别测试过程中的风险，及时采取措施解决规避。

- 场景覆盖结果展现（按部门）:



	T-6周	T-5周	T-4周	T-3周	T-2周	T-1周	T+1周	T+2周	T+3周
场景覆盖进度	12.37%	28.93%	46.12%	56.23%	79.50%	96.87%	100.00%		
测试进度标杆	12.46%	31.30%	52.53%	72.50%	80.50%	95.00%	100.00%		
历史最优测试进度	4.67%	37.02%	79.66%	93.96%	98.80%	98.93%	100.00%	100.00%	100.00%

- 场景覆盖结果展现（按项目）:

项目编号	项目名称	所属应用	开发组	规模	测试小组	2017/2/28							
						有SQL的场景数1	已覆盖场景数1	手工确认场景数2	已覆盖场景数2	总场景数	已覆盖场景	已覆盖场景占比	测试计划进度
20161234	项目A	F-RTC	开发三组	575	测试一组	48	45	0	0	48	45	93.75%	100.00%
20161235	项目B	F-ZFR	开发二组	310	测试一组	32	32	0	0	32	32	100.00%	100.00%
20161236	项目C	F-PCE	开发四组	1678	测试一组	144	139	0	0	144	139	96.53%	100.00%
20161237	项目D	F-SDF	开发一组	523	测试一组	45	45	9	9	54	54	100.00%	100.00%
20161238	项目E	F-IFM	开发一组	45	测试一组	2	2	2	2	4	4	100.00%	100.00%

- 异常覆盖情况分析:

项目编号	项目名称	所属应用	开发小组	规模	测试小组	2017/2/22					异常场景评估
						总场景数	有SQL场景已覆盖	手工确认场景已覆盖	已覆盖场景占比	测试计划进度	
20161162	纽约分行特色功能	F-CCB	开发二组	498.987	测试四组	43	27	7	79.07%	92.31%	疑似重报及合规检查部分案例滞后覆盖，本周追赶完成覆盖
20161162	业务流程优化项目	F-PTE	开发一组	498.987	测试五组	20	13	3	80.00%	92.59%	剩余4个场景均为隔日NAK，需3个日终批量才可完成测试批量需要，本周完成覆盖。
20161614	新增机构外币现钞存取数据项目	F-BOT	开发四组	53.64	测试一组	4	2	0	50.00%	92.86%	测试进度滞后，本周交付前可以覆盖完成。

二、场景覆盖管理的运用

场景三：

组长：“这个项目的验收测试情况怎么样？”

测试经理：“目前场景覆盖进度 87%，高于计划进度，从各应用场景覆盖情况分析，……，测试情况总体正常。”

适应性测试分析：

截止8月19日：测试进度34.38%，时间进度32%；

截止8月24日：测试进度52.53%，时间进度42%；

截止8月31日：测试进度66.37%，时间进度58%；

截止9月7日：测试进度87.04%，时间进度71%；

主应用	否	是	总计	覆盖率
F-RFC	3	107	110	97.27%
F-SCP	16	85	101	84.16%
F-SCP批量	31	91	122	74.59%
F-SCMS	17	114	131	87.02%
F-CCF	0	53	53	100.00%
总计	67	450	517	87.04%

未覆盖场景分析：
SCP应用未覆盖场景：8个场景为CIPS主机报送应用监控，2个一级行合规转退汇，2个分行收SWIFT报文自动转汇CIPS，2个NAK重发，1个差错记录核销，1个注资退汇收报；
SCMS应用未覆盖场景：7个非支付类合规检查场景，6个SCMS收报同步主机状态场景，3个分行发送查询报文，1个跨行报文自动分拣，1个产品层入账转退汇；
SCP批量未覆盖场景：25个日终账务处理场景，5个批量对账场景，1个批量监控场景；
RFC未覆盖场景：1个BAPP发报内扣场景，1个SWIFT发报反交易和1个SWIFT发报特大额授权场景。

上面这个场景就是我们当时做的一个项目的真实的案例，通过场景监控分析，可以清楚的了解到下游测试阶段的测试覆盖情况。场景覆盖管理在系统测试、验收测试测试等阶段已经广泛使用，用于测试进度监控，项目质量分析评估等，通过对场景覆盖情况的统计和分析，客观真实的展现项目测试进展情况，实现了精细化测试过程管理。

三、场景覆盖方法优势

通过几个版本的项目实践，场景覆盖管理已经在部门内部全面推广，广泛用于系统测试和验收测试测试的测试过程管理。我们还把场景覆盖管理作为验收测试的过程管理的一个有效手段，分析测试进展，识别项目风险。场景覆盖之所以广泛使用，总结起来，有几点优势：

1、通过需求追溯，项目团队一起完善场景，确保业务需求、设计方案和测试覆盖保持一致，避免场景测试遗漏；

2、通过技术手段统计，在减少测试人员勾选确认案例工作量的同时还能减少人为偏差，执行结果客观透明，客观真实的反映测试覆盖情况，并适用于各阶段测试环境;

3、场景设计以业务功能模块为基础，便于多维度质量分析和风险评估，使测试过程管理更精细化，有方法可循，不再依赖于测试经理的经验。

四、后续展望

在场景覆盖管理推广使用过程中，我们也遇到过一些问题，比如场景、SQL 计划管理，报表展现等，通过不断的改进完善，优化流程操作，目前已建立一套完整成熟的流程机制，确保每个版本可以有效开展。当然，这个流程机制并非没有改进空间，我们依然在不断的提升，展望未来，仍需不断进步。

1、进一步完善场景设计方法，确保场景覆盖更加客观全面;

2、对场景覆盖数据进一步分析挖掘，从更多维度揭示评估项目研发过程中的质量风险;

3、通过测试驱动开发，促使开发设计时即考虑技术统计的可行性，使场景可监控、易监控，降低场景 SQL 编写维护成本。

测试工作，你必须了解的那些事

◆作者：王 鑫

经常在知乎和软件测试 QQ 群里看到一些刚刚入门或者想要做从事测试职业的人问一些关于测试工作的问题，针对经常会问到的几个问题根据笔者这 5 年的工作经历进行下梳理，希望对大家有所帮助。

1. 测试工作入门

测试入门简单么？个人认为只要你对测试工作有一定了解，有一定的计算机基础，认真准备 2 个月，系统学习测试工作的基础知识，相信你一定会找到一份令你满意的测试工作。

我本人是计算机专业毕业，上学时代的我一心想着从事其他行业工作，对未来没有一点规划，临毕业前看到寝室里的同学都在找测试方面的工作，当时的想法是不用编代码，而且听说非常适合女生，就云里雾里的找起工作来，最后整个寝室就只有我一个人从事这份工作，想想也是很幸运呢。

当然，找到测试工作不难，难的是进阶，对于自己要有明确的目标和规划，对于自己的定位在哪里，测试的路很长，需要学习了解的知识多而杂，需要读着有一定的思想准备。

2. 工作技能要求

纵观各大招聘网站，目前仅仅会黑盒测试已经远远不能满足招聘公司的要求，除了会基本的熟悉黑盒、白盒测试方法，熟悉数据库，编写测试用例之外，以下技能也是需要尽可能掌握的

接口测试

接口测试是功能测试的上游，测试的重点是要检查数据的交换，主要测试这些系统对外部提供的接口，验证其正确性和稳定性。接口测试和功能测试相互配合会做到事半功倍的效果，举个简单的例子，APP 测试时商品智能排序需求设计是商品按照门店由近到远进行排序，然而功能测试时通过测试数据你是无法判断所选商品所在门店的具体位

置，通过和接口测试配合查看数据返回的结果同 APP 显示是否一致，就简化了我们的测试过程。

编程语言

熟悉一门编程语言（java/python/php/c++），如果在你的面试简历上加上这一条，无疑会大大增加面试录取成功的概率。在这里为什么我没有提出性能测试、自动化测试、压力测试，个人认为这些仅仅是使用的工具，核心还是需要懂得代码，万变不离其中就是这个道理。

3. 作为测试新人，我该如何做才能快速成长

上进心为王道

对于这个问题，我深有感触，最近项目组来了两位毕业 1 年左右的新同学，正好这两位同学被分到测试同一个功能模块，平时只完成交代的任务，处于应付的状态，其他功能基本处于不了解的状态。在我看来，是来混工作年限的，毫无上进心。同理，如果你作为 leader，还想继续留用这样的人么

沟通

谦虚的态度，良好的沟通是你和同事互相合作的基础。不懂的问题先自己去摸索，用头脑思考后仍无法解决，再去前辈请教，相信大家会很愿意帮助你的

学习

从工作中学习是最快的成长途径，打个比方来说平时看书 1 一个月所学的知识可以在工作中 1 周甚至几天掌握，而且印象会非常深刻，工作过程中将不懂的地方记下，待不影响进度的情况下摸索或请教他人

及时总结

鱼的记忆只有 8 秒钟，人的记忆能力也是有限的，经常性总结，把自己的知识梳理成文档，一来可以帮助新同学熟悉业务，二来当自己忘记了翻出来看一看会节省很多时间

最后，希望想从事测试工作的同学，心里做好准备，任何一份职业都是神圣的，拿出自己信心、态度、做好规划，人生经不起太多错误的选择及失败，多一份了解，少一份弯路。

自动化测试之 QTP 学习笔记

◆ 作者：含羞草

一、初级 QTP 学习

1、 学习要求

初级阶段的 QTP 学习是一个从无到有的过程，一方面思维需要从原来的手工测试慢慢转变成自动化测试，另一方面对自动化测试的整个流程需要慢慢进行初步认识与学习。具体学习方向包括以下几点:

1）理解一些基础概念：什么是自动化测试，自动化测试能做什么，自动化测试的优势和局限以及如何借助自动化测试工具更好进行自动化测试等等;

2）熟悉 QTP 的界面设置以及一些基本的用法，主要包括脚本录制、脚本回放、参数化、检查点、OUTPUT 输出值等等。具体进行自动化测试的整体流程，每一个流程下的操作方式以及注意事项等等;

3）掌握自动化测试用例的设计技巧，能用脚本对测试用例进行最大覆盖，比较从手工测试到自动化测试在测试用例设计上的共性与差异;

4）可以参照官方学习文档 Tutorial.pdf 对自动化测试进行进一步学习;

二、基本概念

1、 什么是自动化测试

自动化测试是把以人为驱动的测试行为转化为机器执行的一种过程。简言之就是“用程序测试程序”。通常，测试人员在设计了测试用例并通过用例评审之后，需要根据测试用例中描述的规程执行测试，得到实际结果与期望结果的比较。引入自动化测试，是为了节省人力、时间或硬件资源，提高测试效率。

2、 什么项目适合自动化测试

不是所有的项目都适合做自动化测试，实施自动化测试之前需要对软件开发过程进行分析，以观察其是否适合使用自动化测试。具体包括以下条件:

1） 软件需求变动不频繁

对于软件需求变动频繁的软件，测试人员需要根据需求的变动重新调整测试用例以及对应的测试脚本，而测试脚本的维护是需要修改、调试，必要的时候还需要修改自动化测试框架。如果维护的成本过高，自动化测试就是失败的，没必要进行自动化测试，可以选择手工测试。

如果项目中有些模块相对稳定，有些模块需求变动很大，就可以针对不同模块做处理，对相对稳定模块进行自动化测试，而需求变动大的仍采用手工测试。

2） 项目周期长

自动化测试需要经过自动化测试需求确定、自动糊测试框架设计、测试脚本的编写与调试等一系列过程，这些过程需要经历相当长的时间，如果项目周期较短，完成自动化测试的一系列过程的成本较高，相对于手工测试达不到提高效率的目的，也就没必要进行自动化测试。

3） 自动化测试脚本能重复使用

如果花费很长时间开发出自动化测试脚本，但是测试脚本不能重复使用，或者重复使用率低，致使其间所耗费的成本大于所创造的经济价值，进行自动化测试就没什么必要。测试脚本能否重复使用主要从以下几个方面进行考虑: 能否适应不同的系统差异（B/S 架构模式和 C/S 架构模式），能否适应不同的自动化测试工具差异（QTP、WinRunner、Test Partner、SilkTest、AdventNet、AdventNet、Selenium 等等），能否适应不同的自动化测试框架差异。

3、 如何选择自动化测试工具

不同的自动化测试工具有不一样的使用场景，只有了解了自动化测试工具的适用场景，才能根据项目的差异选择合适的自动化测试工具。具体如何选择自动化测试工具，参考以下几个方面:

1）选择尽可能少的自动化产品覆盖尽可能多的平台，以降低产品投资和团队的学

习成本;

2）测试流程管理自动化通常应该优先考虑，以满足为企业测试团队提供流程管理支持的需求;

3）性能测试自动化产品将优先于功能测试自动化产品。

4、 自动化测试能做什么

对于自动化测试能做什么这个问题其实就看我们希望用自动化测试做什么。其实这个问题本身就是一个测试需求设计的问题。我希望大家能接受“测试需求”这个概念，不是只有客户才有需求，我们测试人员也有同样有测试需求，这个需求来源于我们希望改进测试过程、提高测试执行的效率和可靠性，扩大测试的覆盖率。其实功能测试、系统测试、性能测试等都会有测试需求，只是大家可能对这些需求太熟悉，所以反而都忽略了。测试需求就是我们的测试目标，自动化测试的需求就是我们的自动化测试目标，就是我们希望自动化测试做什么。

三、自动化测试工具-QTP

1、 什么是 QTP

QTP 是 Quick Test Professional 的简称，是一种自动测试工具。自动化测试主要是用于回归测试和测试同一软件的新版本，减少手工测试的重复性工作。因此在测试前需要测试人员考虑好如何对应用程序进行测试，例如要测试哪些功能、操作步骤、输入数据和期望的输出数据等。使用自动化测试的目的是为了提高测试效率，如果使用自动化测试没有节约测试成本，就没必要使用自动化测试。

2、 QTP 的特点

1）QTP 是一个侧重于功能的回归自动化测试工具，主要提供了像.NET 的，Java 的，SAP 的，Terminal Emulator 的等等各种用途的插件，分别用于各自类型的产品测试。默认提供 Web，ActiveX 和 VB 插件。

2）QTP 支持的脚本语言是 VBScript，VBScript 是一种松散的、非严格的、普及面很广的语言。这对于测试人员来说，便于学习掌握。

3）QTP 支持录制和回放的功能。QTP 在录制时，对于那些 QTP 不容易识别出来的对象提供一种有用的 lower level 功能。它使用坐标来标识的，对于那些坐标位置频繁变

动的对象，这种方式就不起作用了。对于录制产生的脚本，可以拿来作为自己编写脚本的 template。

4）QTP 的编辑器支持两种视图：Keyword 模式和 Expert 模式。Keyword 模式提供一个描述近似于原始测试用例的、跟代码无关的视图（基本很少用，除了查看、管理当前 test 中各个 action 的完整流程），而 Expert 模式就是代码视图，一般编写脚本都在这个区域。

5）QTP 提供了一个有用的工具：Object Spy，可以用来查看 Run-time object 和 Test object 属性和方法。

6）QTP 通过三类属性来识别对象：a）Mandatory； b）Assistive； c）Ordinal identifiers。大部分情况下，通过对象的一些特定属性值就可以识别对象（类型 a）。这些属性可以通过 Tools->Object Identification 定义。

7）Object Repository（OR）是 QTP 存储对象的地方。测试脚本运行后，QTP 根据测试脚本代码，从这个对象库中查找相应对象。每个 Action 可以对应有一个或者多个 OR，也可以设置某个 OR 为 sharable 的，这样可以供其他 Action 使用。注意，使用 QTP 录制功能时，默认将被测对象放在 local OR 中，可以通过 Resources->Object Respository，选择 Local 查看。

8）说到 QTP 的要点，不得不说 Action。Action 是 QTP 组织测试用例的具体形式，拥有自己的 DataTable 和 Object Repository，支持 Input 和 output 参数。Action 可以设置为 share 类型的，这样可以被其他 test 中的 Action 调用（注意：QTP 是不支持在一个 test 中调用另外一个 test 的，只有通过 sharable action 来调用）。

9）如 4）所述，一个 test 中，多个 action 的流程组织，只有通过 Keyword 视图查看和删除，在 Expert 视图中没有办法看到。

10）调用 Action 可以通过菜单 Insert->Call to *** 来实现。QTP 提供三种类型的调用方式：a）call to new Action，在当前 test 中创建一个新的 Action； b）call to Copy of Action； c）call to existing action，调用一个 re-usable action，如果这个 re-usable action 来自另外一个 test，将以只读的方式插入到当前 test 中。

11）QTP 提供 excel 形式的数据表格 DataTable，可以用来存放测试数据或参数。DataTable 有两种类型：global 和 local。QTP 为 DataTable 提供了许多方法供存取数据，

在对测试代码进行参数化的时候，这些方法非常有用。

12）环境变量（Environment Variables）。在一个 test 中，环境变量可以被当前 test 中所有 action 共享。环境变量也有两种类型：build in 和 user defined。用户自定义的环境变量可以指向一个 XML 文件，这样可以实现在众多 test 之间共享变量。

13）QTP 可以引用外部的 VBS 代码库，通过 Settings->Resource 加入，也可以 ExecuteFile 命令在代码中直接执行。这种 VBS 库可以为所有 action 和 test 共享。

14）QTP 默认为每个 test 提供一个测试结果，包括 Passed，Failed，Done，Warning 和 information 几种状态类型，可以进行对结果 Filter。但是，只能为每个 test 产生一个 testing result，不能为多个 testing 产生一个总的 testing result.

3、 QTP 的优缺点

优点:

1）QTP 针对的是 GUI 应用程序，包括传统的 Windows 应用程序，以及现在越来越流行的 Web 应用。它可以覆盖绝大多数的软件开发技术，简单高效，并具备测试用例可重用的特点。

2）QTP 完全模拟终端用户，独占屏幕，只能开启一个独占的实例;

3）QTP 对 UI 组件以及对 UI 对象的管理存储内置了良好的支持;

4）QTP 脚本创建相当容易，创建脚本的语言采用的是 VBScript，通俗易懂;

缺点:

1）QTP 对系统的环境要求很高，特别是一些类似 360 之类的实时监控软件会起冲突，还有麦克菲杀毒是绝对不能共存的，在使用 QTP 是需要把实时监控软件和杀毒软件关闭，建议在比较空白的机器上跑 QTP;

2）QTP 在运行时会占用很大内存，这对测试机器内存要求很高;

3）QTP 不支持面向对象语言和扩展性支持（和外部工具和库的集成）:

4）QTP 只支持 Windows 系统，不支持跨平台开发;

5）QTP 不支持开发流程的集成;

4、QTP 测试步骤

运用 QTP 进行自动化测试-功能测试的测试流程为：制定测试计划——>创建测试脚本——>增强测试脚本功能——>运行测试——>分析测试结果。具体每一步的步骤如下所示：

1）制定计划

自动测试的测试计划是根据被测项目的具体需求，以及所使用的测试工具而制定的，完全用于指导软件测试全过程。

QTP 是一个功能测试工具，主要帮助测试人员完成软件的功能测试，与其他测试工具一样，QTP 不能完全取代测试人员的手工操作，但是在某个功能点上，使用 QTP 的确能够帮助测试人员做很多工作。在测试计划阶段，首先要做的就是分析被测应用的特点，决定应该对哪些功能点进行测试，可以考虑细化到具体页面或者具体控件。对于一个普通的应用程序来说，QTP 应用在某些界面变化不大的回归测试中是非常有效的。

2）创建脚本

当测试人员浏览站点或在应用程序上操作的时候，QTP 的自动录制机制能够将测试人员的每一个操作步骤及被操作的对象记录下来，自动生成测试脚本语句。与其他自动测试工具录制脚本有所不同的是，QTP 除了以 VBScript 脚本语言的方式生成脚本语句以外，还将被操作的对象及相应的动作按照层次和顺序保存在一个基于表格的关键字视图中。比如，当测试人员单击一个链接，然后选择一个 CheckBox 或者提交一个表单，这样的操作流程都会被记录在关键字视图中。

3）增强脚本

录制脚本只是为了实现创建或者设计脚本的第一步，基本的脚本录制完毕后，测试人员可以根据需要增加一些扩展功能。QTP 允许测试人员通过在脚本中增加或更改测试步骤来修正或自定义测试流程。如增加多种类型的检查点功能，既可以让 QTP 设置检查点检查一下在程序的某个特定位置或对话框中是否出现了需要的文字，还可以检查一个链接是否返回了正确的 URL 地址，还可以通过参数化功能，使用多组不同的数据驱动整个测试过程等等。这些操作都能更加完善脚本的功能，增强脚本的重复使用率。

4）运行测试

QTP 从脚本的第一行开始执行语句，运行过程中会对设置的检查点进行验证，用实际数据代替参数值，并给出相应的输出结构信息。测试过程中测试人员可以根据实际测

试过程调试自己的脚本，直到脚本完全符合测试要求。

5）分析测试

运行结束后系统会自动生成一份详细完整的测试结果报告。测试人员根据测试结果进行分析总结，自动化测试过程结束。

四、QTP 10 安装

1、 前期准备（下载相关软件）

1）下载 QTP10 的镜像文件：QTP10.iso 文件，具体下载地址可以参考如下链接地址：http://www.51testing.com/html/98/n-3717098.html

2）下载破解 QTP10 安装许可证书密钥所需要的应用程序：mgn-mqt82.exe 文件，这个应用程序百度上很多，在这里就不提供下载地址;

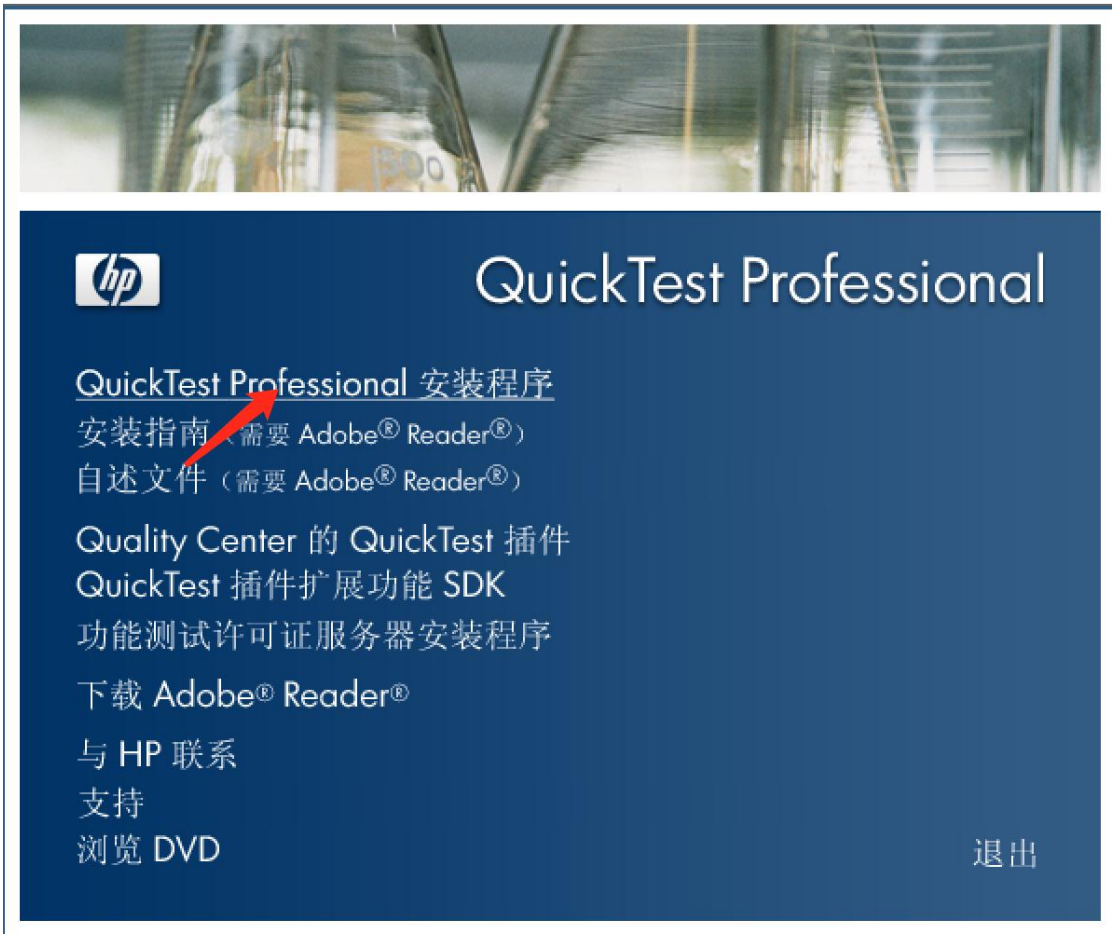
3）下载 QTP 脚本调试器，防止在网络不佳或者断网的情况下 QTP 系统不能自动下载安装 QTP 脚本调试器，导致 QTP 安装不能正常进行。可以参考如下所示的下载地址：http://www.51testing.com/html/00/n-3717100.html

4）将上面这些软件下载好放在一起，方便后期安装和破解 QTP 使用，具体文件如下所示:

2、 安装 QTP10

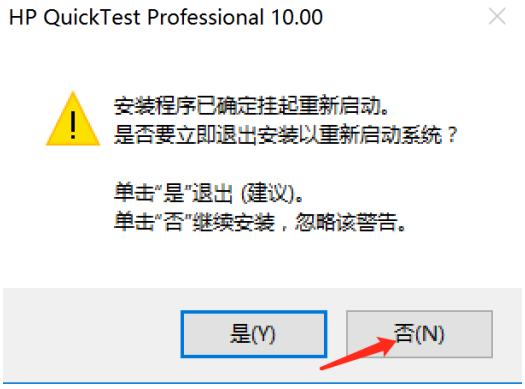
1）双击 QTP10 的镜像文件 QTP10.iso，就能进入镜像文件 ，查看显示的目录，正常显示的目录如下所示:

2）找到应用程序 setup.exe 文件，双击 setup.exe 文件进入 QTP 安装引导页面，点击第一个安装选项：QuickTest Professional 安装程序，进入 QTP 程序安装。具体界面如下所示：



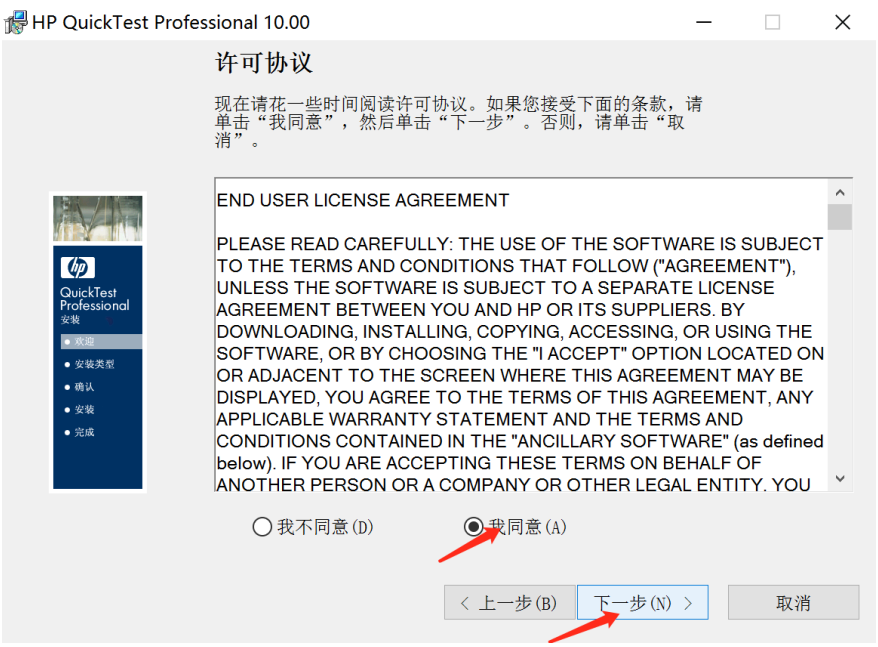
3）此时页面弹出 QTP 安装警告，提示安装程序已确定挂起重新启动，是否要立即

退出安装以重新启动系统，选择是的话退出安装，选择否，继续安装，忽略该警告。这里我们选择否;

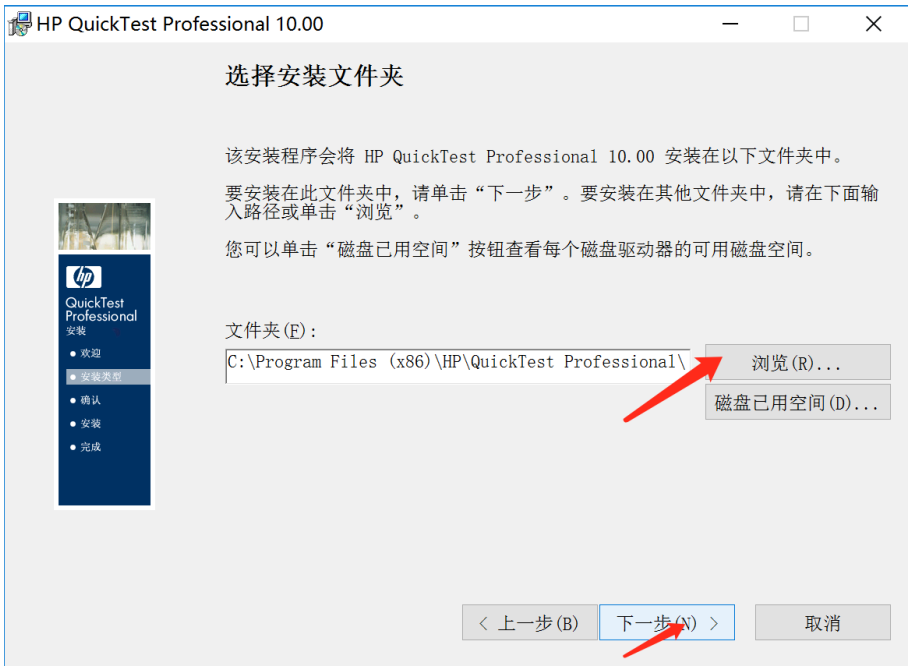


4）安装进入 QTP10 安装向导页面，直接点击下一步;

5）安装进入 QTP10 安装许可协议页面，选择我同意，点击下一步按钮，具体页面如下所示:



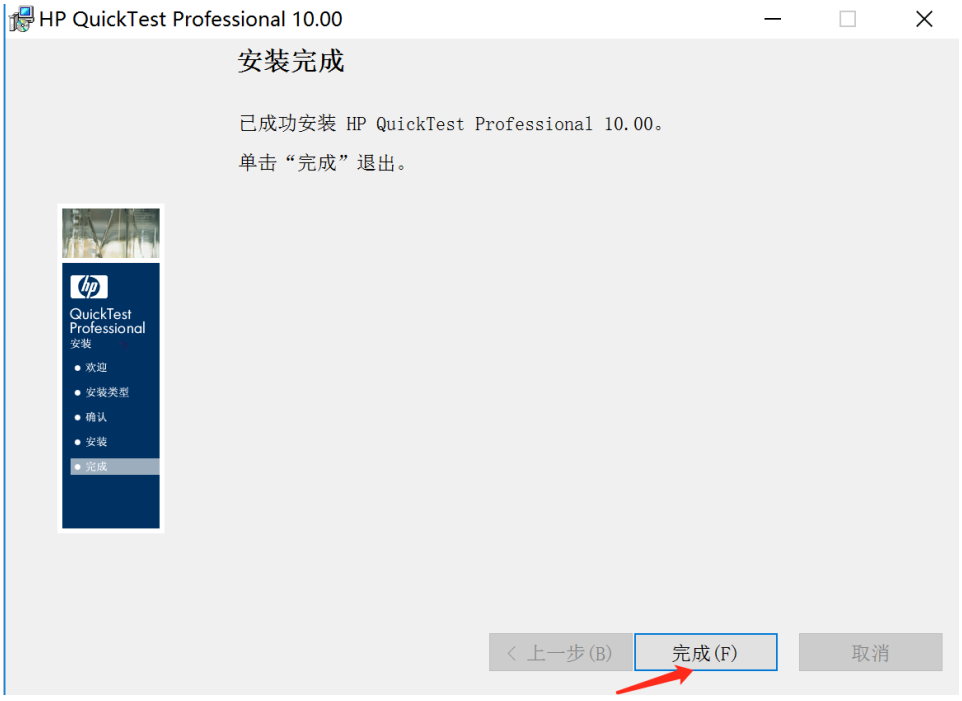
6）安装进入组织填写页面，可以填写组织再点击下一步按钮，也可以不填直接点击下一步按钮。此时安装页面进入选择安装文件夹页面，默认位置安装可以直接点击下一步，如果想要安装在其他文件夹，就单击“浏览”按钮，此时页面弹出窗口可以对安装文件夹进行选择。如果单击“磁盘已用空间”按钮，可以查看每个磁盘驱动器的可用磁盘空间。根据自己的需要以及各个磁盘驱动器的可用磁盘空间选择自己想要安装的文件夹，然后点击下一步按钮。具体界面如下所示：



7）安装进入 QTP 确认安装页面，直接点击下一步按钮，页面进入正在安装 QTP10.00 页面，此时只有取消按钮可以进行点击，其他按钮均置灰，不能点击。可以看

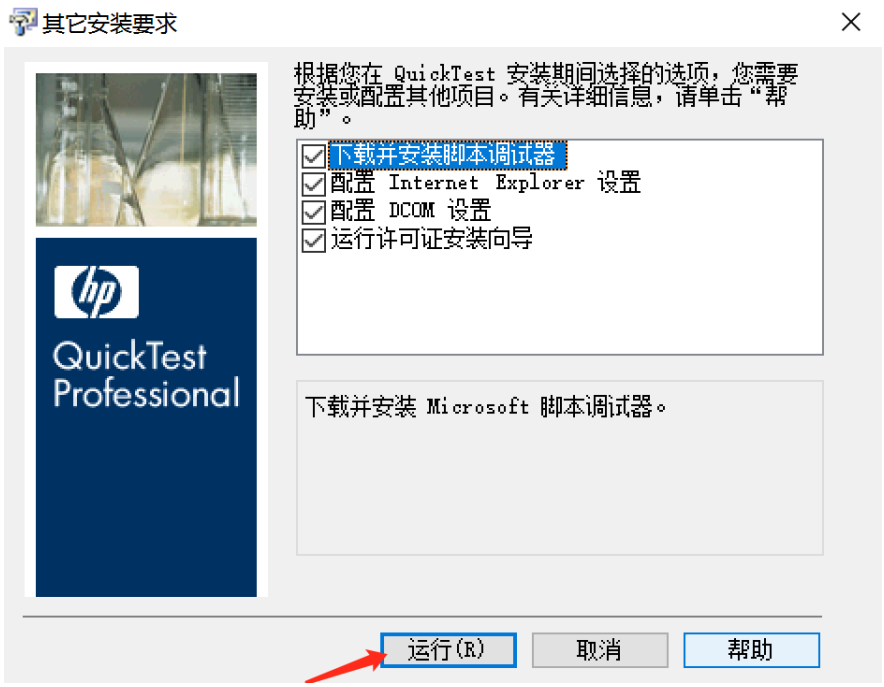
到安装进度条也在不断更新，说明系统正在进行 QTP 验证安装、各个组件安装、注册表安装与更新等等。具体界面如下所示:

8）以上 QTP 安装过程需要持续一段时间，需要我们耐心等待一段时间。等到进度条进行到最后，基础安装基本上完成，此时页面跳转至以下的安装完成页面。直接点击完成按钮，退出安装界面，基础组件安装告一段落。



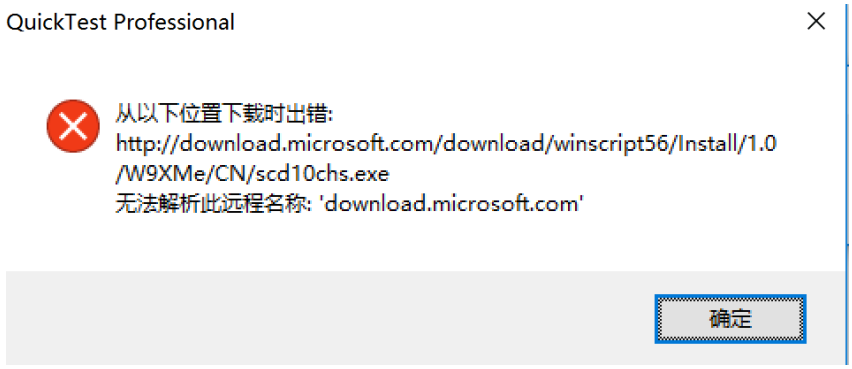
9）基础组件安装完成之后，过一段时间会出现其他安装要求页面，我们还需要进

行其他的一些安装与设置，如下图所示，点击页面上的运行按钮，开始进行安装；



10）点击运行之后，开始进行第一项下载并安装脚本调试器，当前页面出现正在下载.....请稍等的弹框，说明系统开始进行下载脚本调试器，如图所示：

11）等待一段时间，如果出现报错，显示从以下位置下载时出错，无法解析此远程名称等等的信息，说明当前网络不好或者有限制，先点击确定，如果没有报错可以跳过此步；

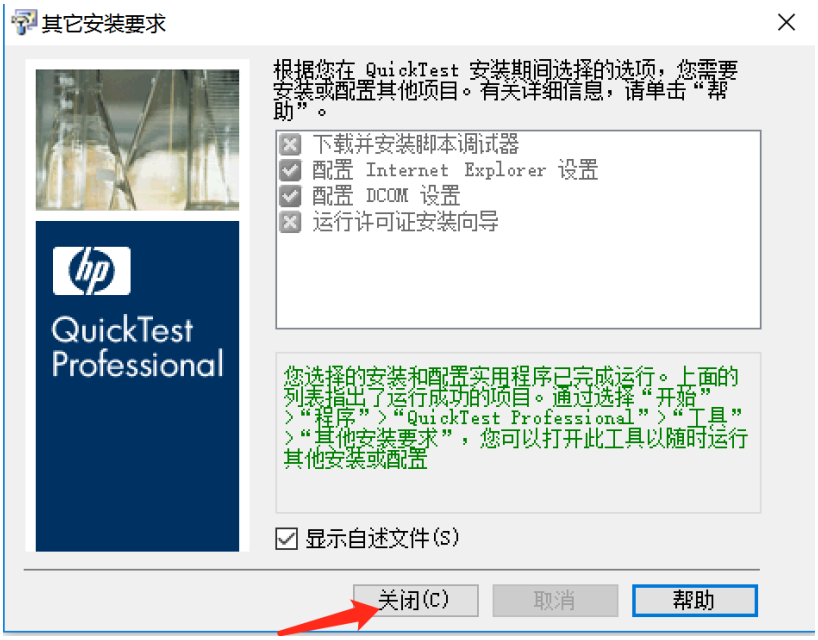


12）出现报错说明脚本调试器不能在安装系统的基础上进行下载，这就需要自己手动下载 QTP 脚本调试器进行安装。

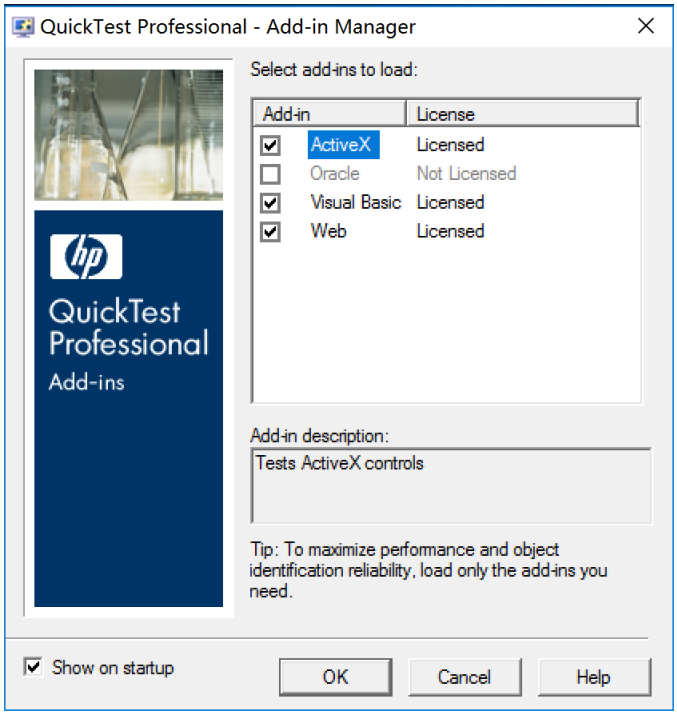
点击确定之后页面跳转至 QTP 许可证安装页面，选择单机许可证，点击下一步;

13）页面跳转至输入许可证密钥页面，目前还没有获取许可证密钥，所以暂时先点击取消，等待后期破解 QTP 获取许可证密钥进行再输入验证;

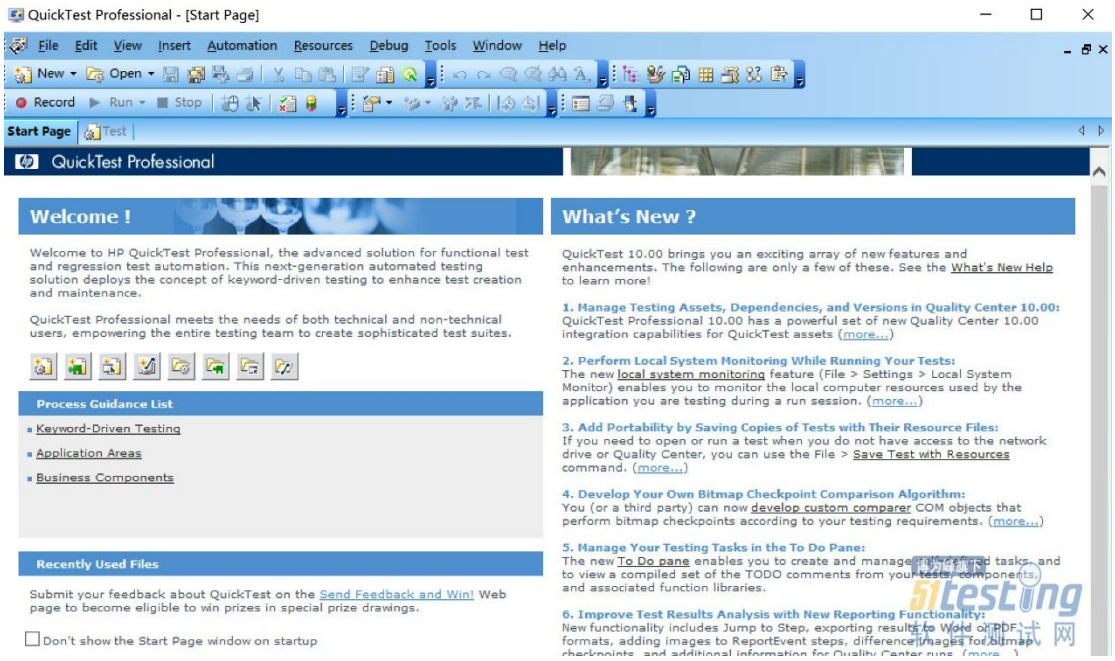
14）许可证安装取消，许可证安装不成功，页面跳转至其他安装要求页面，提示正确安装以及未进行正确安装的项。成功安装的项前面是对勾，没有成功安装的项前面是叉叉。从上面来看，QTP 脚本调试器以及许可证未进行正确安装配置，等待后期解决破解之后可以按照提示进入其它安装要求页面重新进行安装，暂时点击关闭按钮；



15）点击关闭按钮之后，QTP 安装告一段落，桌面会出现 QTP 的快捷方式。双击它，页面进入 QTP 的 Add-in Manager 页面，选择 License 状态为 Licensed 的项，点击 OK 按钮；



16）页面进入 QTP 的欢迎页面，表明 QTP 安装完成。

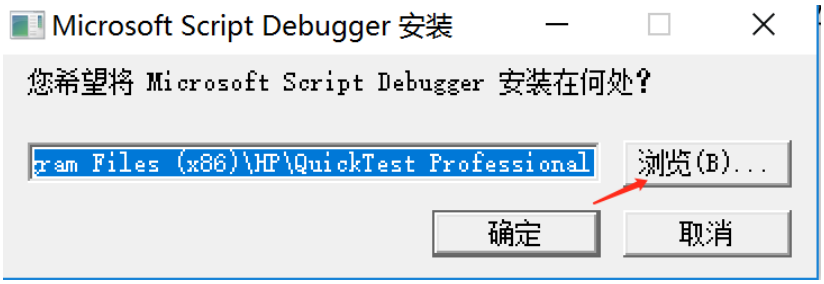


五、QTP 破解

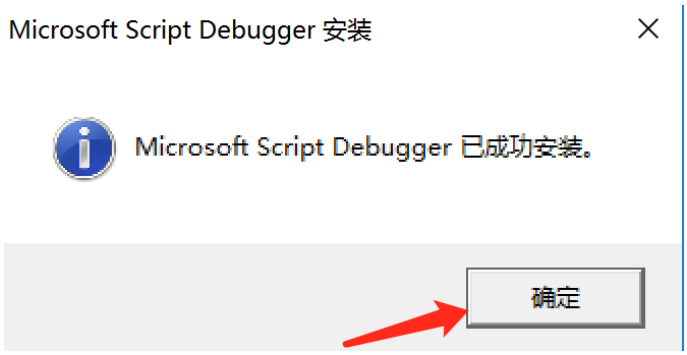
1、 手动安装 QTP 脚本调试器

1）双击脚本调试器.exe 应用程序，页面进入确认安装页面，点击确定按钮;

2）页面进入选择安装路径页面，选择想要安装的位置，点击确定按钮;

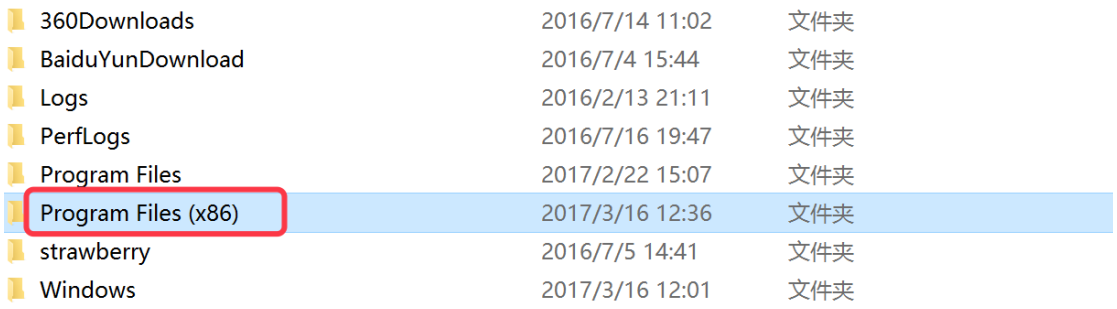


3）页面进行 QTP 脚本调试器安装，过一会提示安装完成，点击确定按钮，完成脚本调试器安装。

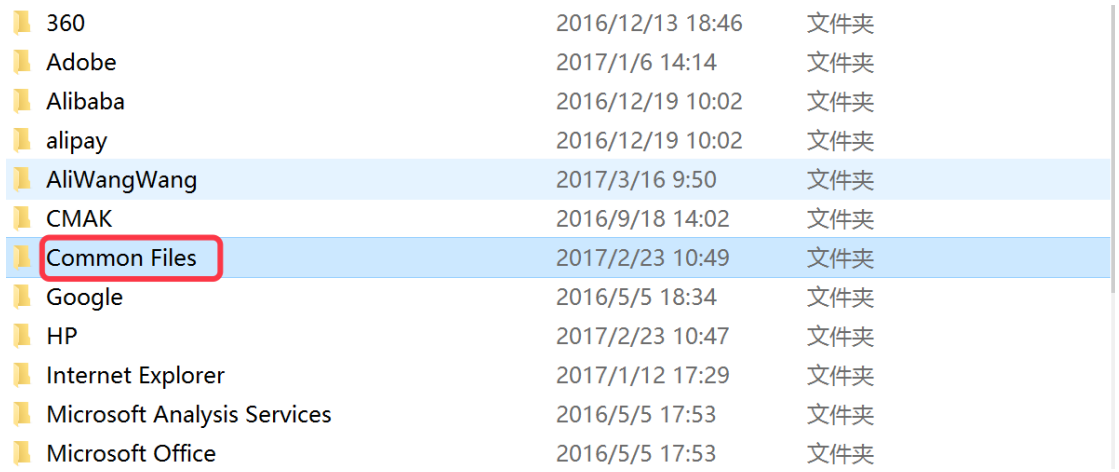


2、 获取许可证密钥

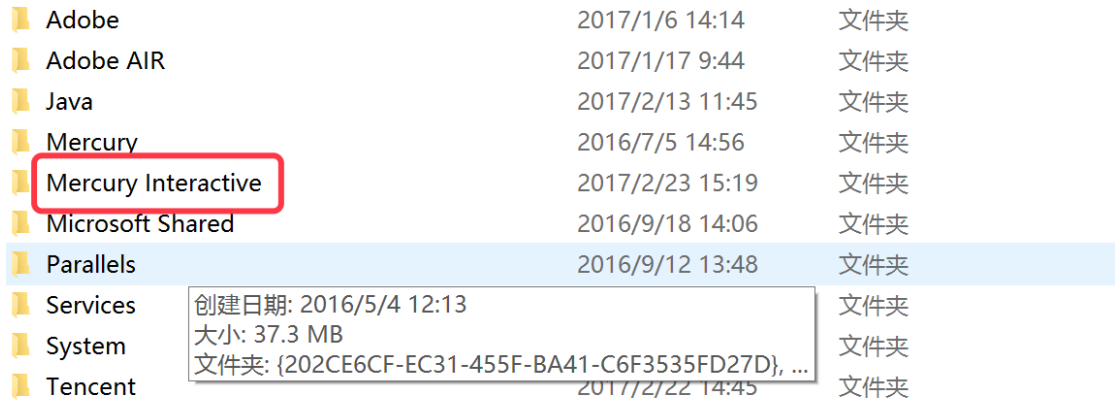
1）进入 QTP 安装的盘符下，根据自己电脑的系统选择相应的系统文件，双击进入系统文件;



2）在系统文件的目录下找到 Common Files 文件夹，双击进入 Common Files 文件夹;



3）从 Common Files 文件夹目录下找到 Mercury Interactive 文件夹，双击进入 Mercury Interactive 文件夹；

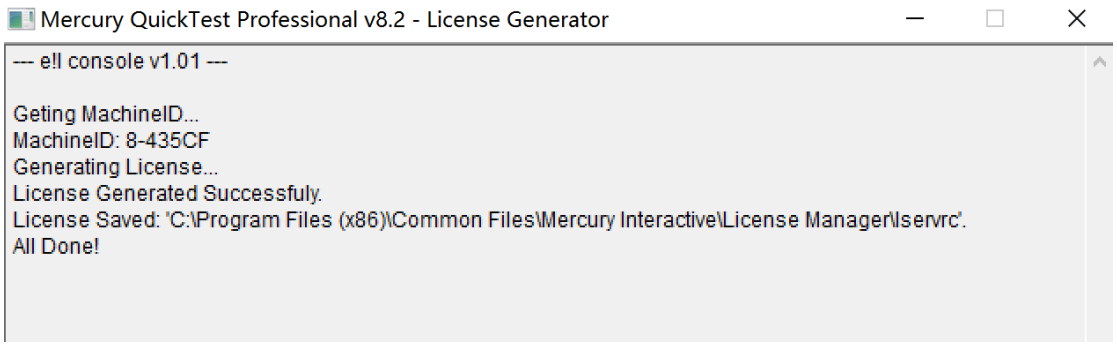


4）查看 Mercury Interactive 目录下是否包含 License Manager 文件夹，如果没有，新建一个 License Manager 的文件夹，如下所示：

5）将前期下载的 mgn-mqt82.exe 应用程序放到 Mercury Interactiv 文件夹下，即跟 License Manager 文件夹在同一目录，如下所示：

6）双击执行 mgn-mqt82.exe 应用程序，页面会弹出相关消息，提示单机的许可证证

书生成成功，以及相应的存放位置。如下所示：

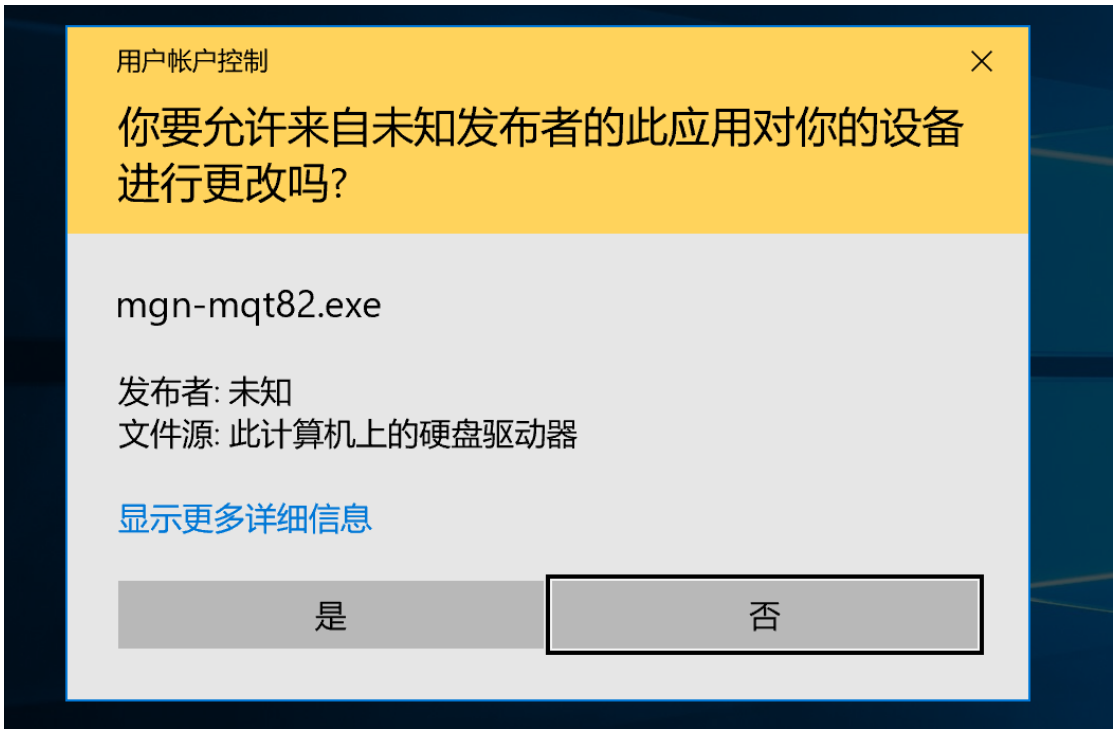


7）查看 License Manager 文件夹下，没有生成 lservrc 文件，说明应用程序没有运行成功，需要更高的权限进行操作。

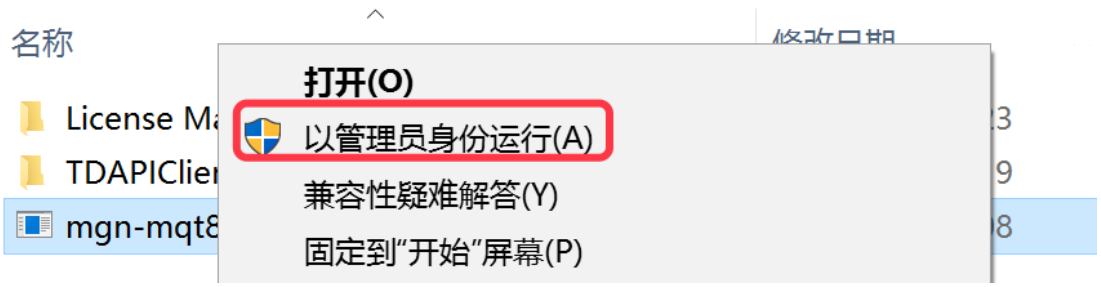
8）设置用户账户为管理员账户：进入控制面板->用户账户->更改账户类型，进入之后，选择管理员，可以将当前用户设置成管理员。设置管理员之后，右击选项会多出以管理员身份运行的选项。

9）更改用户账户控制设置：进入控制面板->用户账户->更改用户账户控制设置，可以控制在 windows 设置改变的时候是否进行提示。

如果将用户账户控制设置成从不提示的话，在以管理员身份运行的时候就不会出现以下窗口，否则会出现该窗口进行进一步确认更改；

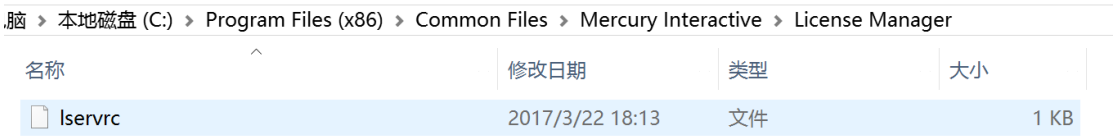


10）右击 mgn-mqt82.exe 应用程序，选择以管理员身份运行。

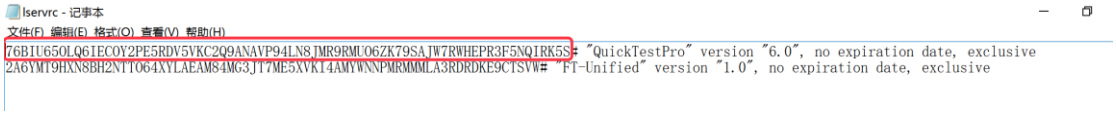


11）同样出现许可证生成成功的提示，这次查看 License Manager 文件夹，出现

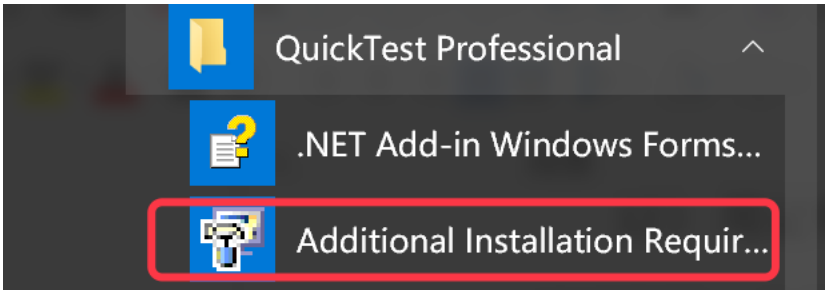
lservrc 文件，说明应用程序运行成功。如下所示：



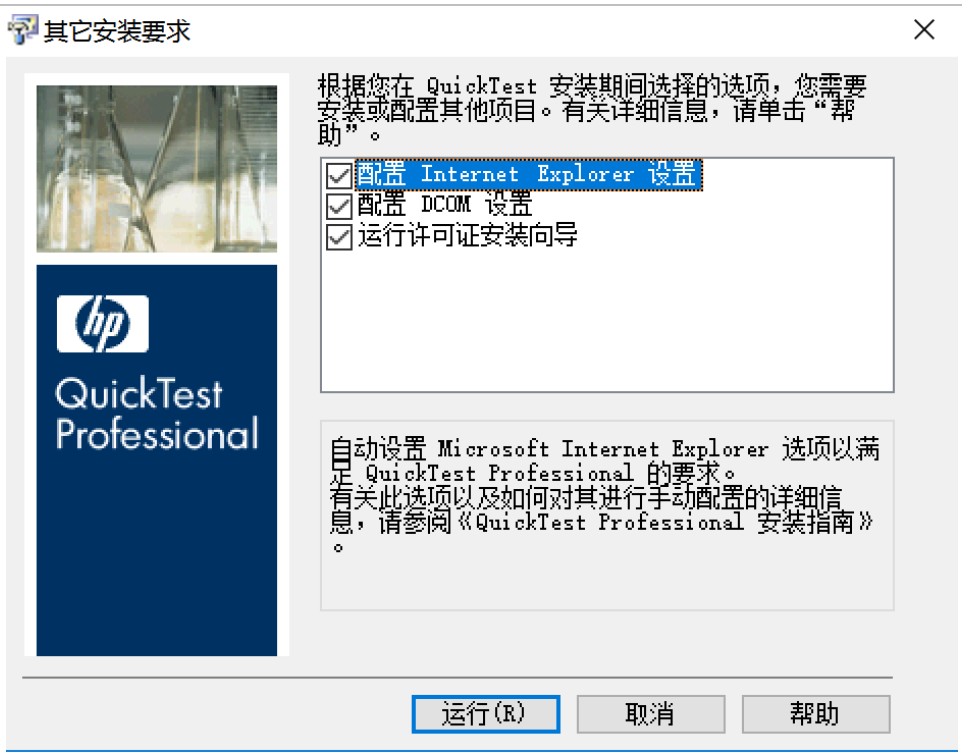
12）用记事本打开 lservrc 文件，#之前的内容就是 QTP 单机许可证密钥，选择#之前的内容进行复制保存，方便后期安装输入。具体哪一些属于许可证密钥如下所示：



13）选择开始>程序>QuickTest Professional>Additional Installation Require ，可以进入其他安装页面；



14）进入其它安装要求页面，此时其它安装要求页面已经没有下载安装 QTP 脚本调试器的任务，因为我们已经进行手动安装成功，直接点击运行；



15）直接点击下一步下一步进入许可证安装页面，在许可证密钥输入框中粘贴入刚刚复制的#之前的许可证密钥，点击下一步，页面显示此单机许可证密钥永久有效，说明 QTP 破解成功，点击完成按钮。

16）单机许可证密钥安装完成之后，页面回到其它安装要求页面，这一次安装与配置任务前面显示的都是对勾，说明所有的安装与配置任务都正常运行。因为单机许可证密钥永久有效，说明 QTP 不会过期，可以永久使用，破解完成，直接点击关闭按钮。

3、 破解验证

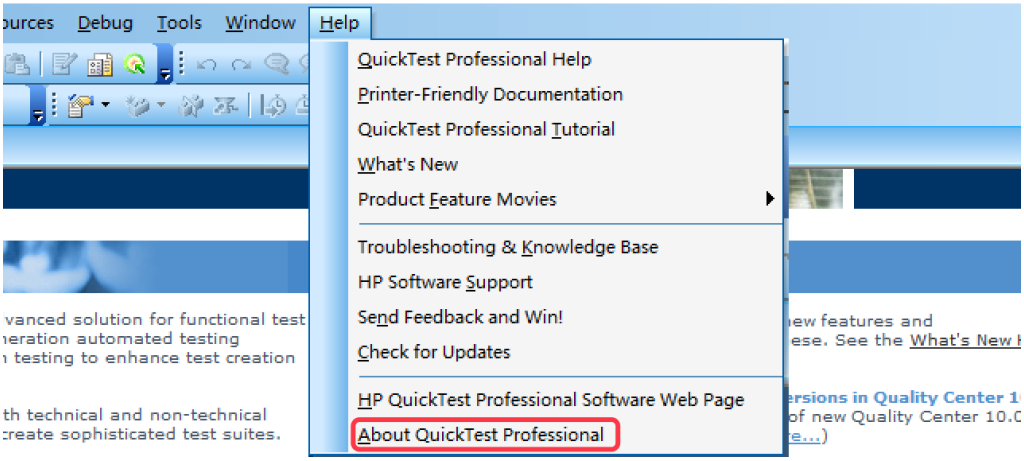
按照以上介绍的方法可以完成 QTP10 的破解工作，使得 QTP 能长期使用。但是破解工作成功与否，就需要我们对 QTP 破解成功与否进行验证。

具体验证方法参照以下方式:

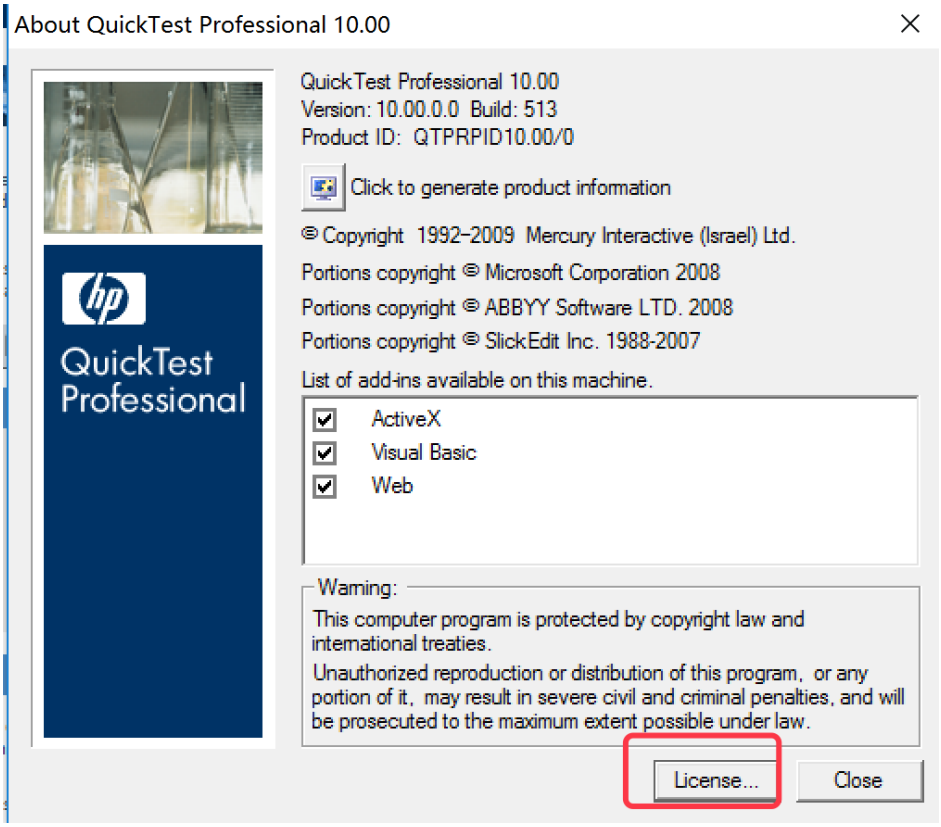
1）双击桌面 QTP 的快捷方式进入 QTP 插件管理页面，直接点击 OK 按钮;

2）页面开始启动 QTP，开始进入 QTP10 的欢迎页面，在页面上方的菜单栏上找到 help 菜单;

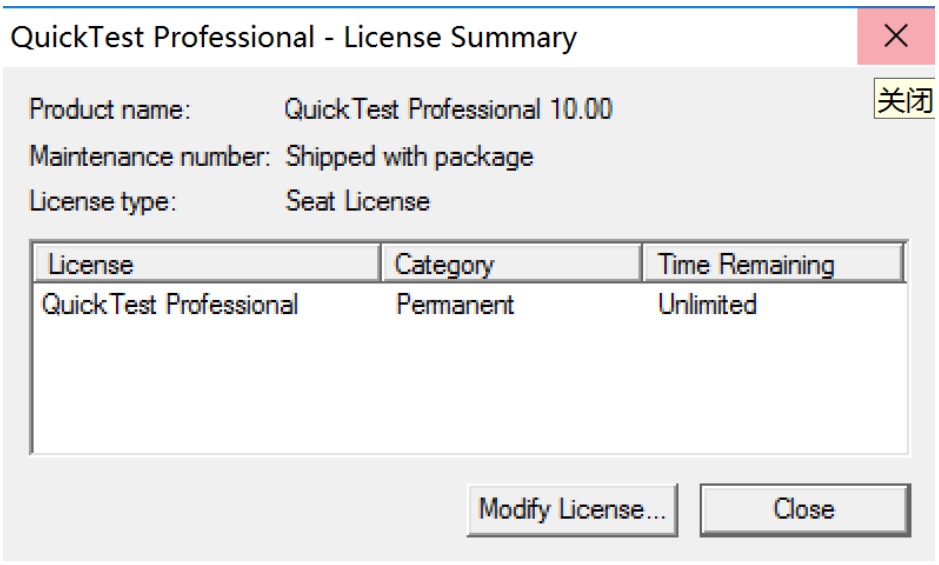
3）点击 help 菜单，会弹出许多选项，点击最后一个 About QuickTest Professional 选项，具体选项显示如下图所示:



3）页面进入 About QuickTest Professional 选项的详细页面，在这里我们可以看到 QTP 的版本、产品 ID 以及机器上的插件等等，点击 License 按钮，具体界面如下所示：



4） 页面进入 QuickTest Professional-License Summary 页面，页面如果出现注册证书且 Time Remaining 的值显示的是 Unlimited 的话说明破解成功，单机证书永久有效。具体界面如下所示：



4、 注意事项

1）手动安装 QTP 脚本调试器这一步操作只在前期安装的时候没有进行正常安装脚本调试器才需要，如果前期已经安装跳过这步操作;

2）进入 QTP 的安装盘符找到系统文件要根据自身系统的位数来选择相应的文件夹，如果是 32 位系统选择 Program Files 文件夹，如果是 63 位系统则选择 Program File（x86）文件夹;

3）Mercury Interactive 目录下若没有 License Manager 这个文件夹需要手动创建，且文件夹名称不能发生错误，否则会影响后期许可证密钥的生成;

4）运行 mgn-mqt82.exe 程序必须以管理员的身份进行操作，否则尽管运行了 mgn-mqt82.exe 程序，系统虽然提示生成了 lservrc 文件，但是在 License Manager 文件夹找不到该文件 lservrc 文件;

5）单机许可证密钥是#之前的内容，不要将所有的内容都拷贝进行粘贴，否则许可证密钥安装不成功;

6）单机许可证密钥是根据机器生成的，有唯一性，不管运行几次 mgn-mqt82.exe 应用程序，生成的单机许可证密钥只有一个;

7）单机许可证密钥一旦安装就有效，就算卸载了 QTP，再次安装的时候，许可证密钥默认已经安装，不需要重复安装。

5、 QTP 安装与破解总结

QTP 是一种自动化测试工具，它的安装与破解过程非常简单快捷。除了按照正常的安装引导进行安装，只要注意单机许可证密钥的获取以及一些注意事项就能正常破解 QTP，使得 QTP 这款自动化测试软件能长期使用。

QTP 的破解也分版本，在 10 以下（包括 10）的版本能正常永久使用，10 以上的版本不能永久破解，破解一次只能使用一个月，到期之后需要再次进行破解。所以本安装破解方法仅适用于 10 以下（包括 10 版本）的 QTP 版本，参考该文档的伙伴要注意进行选择使用。更高版本的安装与破解需要参考其他文档，希望测试伙伴能理解。

《51 测试天地》(四十五) 下篇 精彩预览

- 解读日常工作中开发和测试的关系
- 接口测试之 Jmeter 数据+关键字驱动实践
- 当测试 APP 的消息推送通知时考虑用户体验性
- 我在国企做测试的日子
- Web 前端性能测试平台开发(Flask)
- Python 自动化测试番外篇-接口测试
- 如何查看日志以及根据日志来构造数据？
- 沟通渠道规范要求
- 从烧烤摊主到测试主管只用了 4 年，过程却曲折离奇
- 以男性思维获取测试需求，以女性思维设计测试用例
- 怎样更好地做好测试工作？
- 软件测试中的系统网络图

马上阅读